305-EMD-001

EOSDIS Maintenance and Development Project

Release 7.11 Segment/Design Specifications for the EMD Project

Revision 02

July 2006

Raytheon Company
Upper Marlboro, Maryland

This page intentionally left blank.

Release 7.11
Segment/Design Specifications for the EMD Project

Revision 02

July 2006

Prepared Under Contract NAS5-03098
CDRL Item #023

RESPONSIBLE ENGINEER

Natisha Greenway 7/26/2006

Natisha Greenway — Date
EOSDIS Maintenance and Development Project

SUBMITTED BY

7/26/06

Art Cohen, Custom Code Maintenance — Date
EOSDIS Maintenance and Development Project

Raytheon Company
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document is a formal contract deliverable. It requires Government review and approval within 45 business days. Changes to this document will be made by document change notice (DCN) or by complete revision.

Any questions or proposed changes can be addressed to:

Data Management Office
The EMD Project Office
Raytheon Company
1616 McCormick Drive
Upper Marlboro, MD 20774-5301

Revision History

Document Number	Status/Issue	Publication Date	CCR Number
305-EMD-001	Original	July 2004	04-0371
305-EMD-001	Revision 01	July 2005	05-0329
305-EMD-001	Revision 02	July 2006	06-0342

This page intentionally left blank.

Abstract

The Release 7.11 Segment/Design Specification is an overview description of the ECS Project. The functionality of the ECS software is described at the Subsystem, Computer Software Configuration Item (CSCI), Computer Software Component (CSC), and Process levels. Architecture and context diagrams illustrate the process interconnections within the ECS CSCIs and the external connections to other CSCIs, subsystems, and specified segment interfaces. Interface event description tables describe the data, messages, notifications, or status information that occurs at each level of functionality within the ECS. A basic description of the Commercial Off The Shelf (COTS) software and hardware used in ECS is included. Also, a more detailed class level of documentation is offered from the output of the on-line documentation tool ABC++.

The high-level design in this document is the level of information derived from requirement sources, and used by the development team to complete the ECS design implementation for a software system at a 7.11 state of maturity.

Keywords: Release 7.11, Overview, SDPS, CSMS, Design, Detailed Design, Subsystem, Architecture, Software, Hardware, Object Oriented, Security, Gateway, System Management, Reports, User Interface and GUI.

This page intentionally left blank.

Contents

4. Subsystem Description

5. Limitations of Current Implementation

List of Figures

List of Tables

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Purpose and Scope

The purpose of the Segment/Design Specification for the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS) is to provide an overview of the hardware and software subsystems of the project. This document describes the high-level design of each ECS software subsystem implemented to satisfy the allocated and derived functional and performance requirements. This document also provides basic descriptions of the Commercial Off The Shelf (COTS) hardware and software used in the ECS. This document contains :

- Functional overviews of each Computer Software Configuration Item (CSCI)
- Context diagrams of each CSCI
- Interface event descriptions based on the context diagrams
- Process architecture diagrams
- Interface event description tables based on the process architecture diagrams
- CSCI data stores (databases as they relate to the process architecture diagrams)
- CSCI functions allocated to processes. For data servers, this includes descriptions of the functionality offered to clients via the server interfaces. For Graphical User Interface (GUI) applications, it describes the functionality provided to the GUI users
- Specific limitations of the capabilities provided
- Summary of object classes listed by CSCI
- Summary of class libraries listed by CSCI
- Abbreviations and Acronyms

1.2 Document Organization

The remainder of this document is organized as follows:

- Section 2: Related Documentation
- Section 3: System Description
- Section 4: Subsystem Description
- Section 5: Limitations of Current Implementation
- Abbreviations and Acronyms

This page intentionally left blank.

2. Related Documentation

2.1 Parent Documents

The parent documents are the documents from which the scope and content of this Design Specification are derived. These documents are listed below.

423-46-01	EMD F&PRS
423-46-03	EMD Task 101 Statement of Work
194-207-SEI	System Design Specification for the ECS Project

2.2 Applicable Documents

Refer to the 900 Series documentation found on the EMD Baseline Information System (EBIS) website: http://cmdm.hitc.com/baseline/.

2.2.1 Other Related Documents and Documentation

205-CD-004	Science User's Guide and Operations Procedures Handbook (Release B.0) for the ECS Project
311-EMD-006	Subscription Server Database Design and Schema Specifications for the EMD Project
313-EMD-001	Release 7.11 ECS Internal Interface Control Document for the EMD Project
333-EMD-001	Release 7 SDP Toolkit User's Guide for the EMD Project
611-EMD-001	Release 7 Mission Operations Procedures for the EMD Project, Section 3.2
625-EMD-004	Training Material Volume 4: System Administration (System Startup and Shutdown section)
193-801-SD4	PGS Toolkit Requirements Specification for the ECS Project (a.k.a. GSFC 423-16-02)
193-WP-118	Algorithm Integration and Test Issues, White Paper for the ECS Project
194-WP-925	Science Software Integration and Test White Paper for the ECS Project
423-16-01	Data Production Software and Science Computing Facility (SCF) Standards and Guidelines

423-42-06	Interface Control Definition for the EOS Data Gateway (EDG): Messages and Development Data Dictionary V0 and ASTER/ECS Message Passing Protocol Specification
RFC 793	Transmission Control Protocol
RFC 768	User Datagram Protocol
RFC 791	Internet Protocol
RFC 1597	Address Allocation for Private Internet
	WWW page is http://cmdm.east.hitc.com

2.3 Information Documents Not Referenced

The documents listed below, while not directly applicable, help in the maintenance of the delivered software.

423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System Core System
423-41-03	EOSDIS Core System Contract Data Requirements Document
230-TP-002	Interface Control Document Between the ECS and the Product Distribution System Information Server (PDSIS)
505-41-21	IRD between ECS and NASA Institutional Support System (NISS)
505-41-30	ICD between ECS and Version 0 System for Interoperability
505-41-34	ICD between ECS and ASTER Ground Data System
505-41-38	ICD between ECS and EOS-AM Project for AM-1 Spacecraft Analysis System
505-41-40	ICD between ECS and the GES DAAC
423-41-57-1	ICD between ECS and SIPS, Volume 1: ECS ACRIM III SIPS Data Flows
423-41-57-2	ICD between ECS and SIPS, Volume 2 SAGE III SCF Data Flows
423-41-57-3	ICD between ECS and SIPS, Volume 3 ASTER OSF
423-41-57-4	ICD between ECS and SIPS, Volume 4 ASTER DEM
423-41-57-5	ICD between ECS and SIPS, Volume 5 MOPITT Data Flows
423-41-57-6	ICD between ECS and SIPS, Volume 6 MODIS (MODAPS
423-41-57-7	ICD between ECS and SIPS, Volume 7 ASMR-E

423-41-57-8	ICD between ECS and SIPS, Volume 8 MLS Data Flows
423-41-57-9	ICD between ECS and SIPS, Volume 9 MTMGW
423-41-57-10	ICD between ECS and SIPS, Volume 10, TES Data Flows
423-41-57-11	ICD between ECS and SIPS, Volume 11, ICESat Data Flows
423-41-57-12	ICD between ECS and SIPS, Volume 12, HIRDLS
423-41-57-13	ICD between ECS and SIPS, Volume 13, OMI
423-41-57-14	ICD between ECS and SIPS, Volume 14, SORCE
423-41-58	ICD between the ECS and LP DAAC
423-ICD-EDOS/EGS	ICD between EDOS and EGS
423-EDOS-0211.0001R2	Interface Requirements Document between EDOS and EGS Elements

2.4 ECS Tool Descriptions

2.4.1 Rational Rose

The Rational Rose tool provides support for object-oriented analysis and design. In particular, the Rose tool provides support for controlled-iterative or component-based development. The Rose tool is used on the ECS Project to document the object-oriented elements of the design using class diagrams, use-case diagrams, interaction diagrams, component diagrams, and object diagrams. The Unified Modeling Language (UML) is the methodology used on the ECS Project for all design activities (although the Rose tool also supports the Booch ’93 Methodology or the Object Modeling Technique (OMT) as well).

The Rose tool can also be used to reverse engineer code developed that lacks supporting documentation to get as-built object diagrams.

Before using the Rational Rose tool, see “Rational Rose 98, Using Rose” for important tool usage and reference information. In addition, the following references can be obtained and used:

(1) “Unified Method for Object-Oriented Development,” by Grady Booch and Jim Rumbaugh (version 1.1, Rational Software Corporation) for an introduction to the respective method’s notation, semantics, and process for object-oriented analysis and design.

(2) the second edition of “Object-Oriented Analysis and Design with Applications” by Grady Booch, (Benjamin/Cummings, 1994)

(3) “Object-Oriented Modeling and Design” by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy and William Lorensen, (Prentice-Hall, 1991)

(4) “UML Distilled: Applying the Standard Object Modeling Language” by Martin Fowler with Kendall Scott, Foreword by Grady Booch, Ivar Jacobson, and James Rumbaugh (Addison Wesley Longman, Inc., 1997)

2.4.2 Clearcase Baseline Manager Configuration Management Tool

ClearCase Baseline Manager (CBLM) consists of the ECS baseline data that it manages and the Graphical User Interface (GUI) used to manipulate the ECS baseline data.

The data comes from three sources:

1) Existing XRP-II data
2) Existing Release Notes (914-TDA-xxx) Machines Impacted data
3) Existing CCR data

The import of the XRP data into ClearCase was performed in several steps. First, the hardware document, 910-TDA-003, was copied as a text file. Each of the items contained within this report is identified with a Control Item Identifier (CID). This is an 8-digit integer with a "b" prefix (e.g., b00083456). Each COTS S/W product has its own CID. Because CIDs are mapped to ECS hosts, it was decided to represent information within ClearCase as elements. Each CID is then a ClearCase element. For the ClearCase CID format, the comma separated variable (CSV) format was chosen, as this format is easily ported into and from other COTS S/W products, specifically Microsoft Access and Excel.

The ClearCase configuration specification chosen was the simplest, or the default configuration specification. A view, CM_MASTER, was created with the default configuration specification to manage the data records. The CID records (checked in ClearCase elements) are located in the /ecs/cm VOB at /ecs/cm/CIDs. This directory currently contains the 256 records that correlate to XRP-II's COTS S/W CIDs.

Another important data construct within CBLM is the notion of the Machines Impacted file, and a CCR identified construct, which maps CIDs to hosts. Each Configuration Change Request (CCR) affecting the baseline contains information about 1 or more CIDs. Also, the CCR contains information regarding the hosts receiving the COTS S/W (CID). So the CCR has a construct that in its simplest form is one "CID_MAP" file, and one Machines Impacted (MI) file. The "CID_MAP" file is a simple lookup table. In this case, there is only one entry. The entry contains first a valid CID, followed by one or more blanks, then the name of the "MI" file. In this case, the MI filename is "MI." The MI file, contains a list of valid ECS hosts having the COTS S/W identified within the CID. So a CCR (03-1234) to place a COTS S/W (e.g., Acrobat Reader), onto host e0acs03, would have an MI file containing one host, e0acs03, and one CID_MAP. If the Acrobat Reader software is CID b00081234, the CID_MAP file would contain:

"b00081234 MI"

And the MI file contains:

"e0acs03"

The CCR would be found at:

/ecs/cm/CM/2003CCRs/1234/, a directory

Under this directory is found the two files, “CID_MAP” and “MI.” Note that there is always only one CID_MAP file for each CCR, but that the CID_MAP can contain more than one line. The simplest example of this is when a COTS S/W product needs to be mapped to both SGI and Sun hosts. Then there would be two MI files, “MI_SGIs” and “MI_Suns”, for example. The CID_MAP would contain two entries, one mapping the SGI hosts to the SGI CID, and the other mapping the Sun hosts to the Sun CID.

As CCRs are required to change the CBLM data state, the effectivity date is then defined as the CCR approval date. This is the date the change becomes valid. The next construct, named the “Sequencer”, is the table providing the basis for change. It has a temporal component; the last approved CCR is at the end of the table. As new CCRs get approved, they simply get concatenated to the end of the list in time order. The Sequencer is an executable script.

The last construct is the “dartboard.” Conceptually, the “dartboard” is a directory within ClearCase, at /ecs/cm/BLM/dartboard/. All ECS hosts are listed as files in the /dartboard/ directory. In conclusion, then data constructs are:

CIDs

CCR directories

CID_MAPs and Mis under the CCR directories

Sequencer

Dartboard

The way these pieces all work together is now discussed.

When a CCR is approved that affects the baseline, a CCR is checked out. The /ecs/cm/CIDs/ ClearCase directory is checked out. The new CID is created and populated with the information present on the CCR form. The new CID number then has a ClearCase element created, and the first version becomes this new CID. The /ecs/cm/CIDs/ directory is then checked back in. Next, the MI file must be prepared. Within the CCR directory, two new files are “made” (cleartool mkelem –eltype text_file –nc CID_MAP MI). The hosts, which are to get the COTS S/W, are entered into the MI file, then the file is checked in as the first version. Next, the CID_MAP file is created, mapping the new CID number to the MI file. The CID_MAP file is checked in, then the CCR file is checked back in. This work gets the CCR information locked into ClearCase.

Next, the Sequencer file is edited to show the new CCR number at the end. This action allows the CCR’s MI and CID_MAP files to overlay onto the ClearCase baseline. This is accomplished by echoing the contents of the CID (in file /ecs/cm/CIDs/b00083123) onto each of the hosts specified with the /ecs/cm/CM/2003CCRs/1234/MI file. This data is written to the hosts files with the dartboard, located at /ecs/cm/BLM/dartboard.

Once the data has been applied to the dartboard, subsequent scripts then produce the output reports. In conjunction with the current hosts list, the scripts obtain all of the valid hosts of the site, and basically reformat the data within the dartboard files into reports, which are XRP like in format. Information was added to the reports, including the CCR number, related Release Notes documentation, and the CCR approval date.

The reports are written to the /ecs/cm/BLM/reports directory. Subsequent “Expect” scripts then ftp those reports to specific locations on the EBIS server, Pete, and then are transmitted over to the public server, cmdm.

The languages used in this tool are “csh”, “expect”, and C. Also, “.grp” files are used to enable the ClearCase GUIs. These files are text files that are dynamically generated at the time that the GUI is launched. Code has been reused from two sources, the DeliveryTool, which is used to prepare and send data to the sites, and the cmdm_replication scripts, which are used to replicate data from the Landover EBIS server Pete, to the public (DAAC, ESDIS) server, cmdm.east.hitc.com.

3. System Description

3.1 Mission and Release 7.11 Objectives

The Mission of the National Aeronautics and Space Administration's Earth Science Enterprise is to develop a scientific understanding of the total Earth System and its response to natural or human-induced changes to the global environment to enable improved prediction capability for climate, weather and natural hazards. The vantage point of space provides information about Earth's land, atmosphere, ice, oceans and biota that is obtained in no other way. Programs of the enterprise study the interactions among these components to advance the new discipline of Earth System Science, with a near-term emphasis on global climate change. The research results contribute to the development of sound environmental policy and economic investment decisions.

The Earth Observing System Data and Information System (EOSDIS) Core System (ECS) has been designated as the ground system to collect, archive, produce higher-level data products and distribute data for the Earth System Science mission.

3.1.1 Release 7.11 Capabilities

The ECS capabilities have been developed in increments called formal releases. Release 7.11, which is managed by Configuration Management, is a formal release. It is a collection of new and updated capabilities provided to the users of the system and is described here to show the progress of system enhancements. The ECS collects and stores, processes, archives and distributes scientific data from six different platforms (satellites). In the following sub-sections, the platforms and instruments from which scientific data is collected are identified, the type of data ingested and archived is presented, search and order capabilities for scientific data, how data is distributed and processed, system architecture and operation, system security and Distributed Active Archive Center (DAAC) and external system support are described. Other capabilities provided by Release 7.11 include processing the data obtained, distributing raw or processed data as requested, quality assurance of processed data, supporting communication networks, and systems monitoring via interfaces with the ECS operations staff.

Release 7.11 unique capabilities and modifications include:

- MD5 Checksum- A new checksum type, MD5 checksum, has been added to the existing end-to-end checksumming capability that is used to ensure data integrity.
- OMS-only distribution of physical media – although the distribution of physical media via OMS was a capability provided in Release 7.10, the existing PDS could still be used for this purpose. In Release 7.11, PDS has been eliminated from the system and physical media can only be distributed via OMS.

3.1.1.1 ECS Support of Instruments by Platform

- The Meteor 3 platform supports the Stratospheric Aerosols and Gas Experiment III (SAGE III) instrument
- The ACRIMSAT platform supports the ACRIM III experiment
- The Terra (AM-1) platform supports the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER), Multi-Angle Imaging SpectroRadiometer (MISR), Moderate Resolution Imaging SpectroRadiometer (MODIS) and Measurements of Pollution in the Troposphere (MOPITT) instruments
- The Aqua (PM-1) platform supports the Moderate Resolution Imaging SpectroRadiometer (MODIS), Atmospheric Infrared Sounder (AIRS), Advanced Microwave Scanning Radiometer (AMSR), Humidity Sounder for Brazil (HSB) and Advanced Microwave Sounding Unit (AMSU) instruments
- The Ice, Cloud and Land Elevation satellite (ICESat) platform supports the Geoscience Laser Altimeter System (GLAS) instrument
- The Solar Radiation and Climate Experiment (SORCE) platform supports the Total Irradiance Monitor (TIM), Solar Stella Irradiance Comparison Experiment (SOLSTICE), Spectral Irradiance Monitor (SIM) and XUV Photometer System (XPS) instruments
- The AURA platform supports the High-Resolution Dynamics Limb Sounder (HIRDLS), Microwave Limb Sounder (MLS), Ozone Monitoring Instrument (OMI) and Tropospheric Emission Spectrometer (TES) instruments

3.1.1.2 Ingest and Archive Capabilities

The following data is ingested and archived in the ECS from the various instruments described in section 3.1.1.1:

- Ingest of science and engineering data from the EOS Data and Operations System (EDOS)
- Ingest of Product Generation Executable (PGE) software from Science Computing Facilities (SCFs) either electronically or via media tape
- Ingest of ASTER Level 1A/1B data from the ASTER GDS
- Ingest of FDS (formerly FDD) orbit data via polling without DR
- Ingest of SAGE III MOC Level 0 data into the Science Data Server
- Ingest of SAGE III SCF higher-level products into the Science Data Server via the SIPS interface
- Ingest of Data Assimilation System (DAS) HDF-EOS data via standard polling with DR
- Ingest of MODIS higher-level products via the SIPS Interface after the MODIS L1A, L1B and ancillary data are provided to the MODAPS SCF from the ECS archive and processed by the MODAPS SCF

- Ingest of MOPITT SCF Level 0 data via the SIPS interface
- Ingest of SDPS resident data across a mode in the same DAAC or across DAACs
- Ingest of ASTER DEM granules (HDF-EOS data and metadata) into the Science Data Server via the SIPS interface
- Ingest of ACRIM Level 0 and higher-level data from the ACRIM SCF via the SIPS interface
- Ingest of higher-level AMSR data products from the AMSR SCF
- Ingest of SORCE Level 0 data from the SORCE Team via the SIPS interface
- Archive of ICESat GLAS Level 1, Level 2 and Level 3 and ancillary data at the NSIDC DAAC
- Archive of SORCE TIM/SOLSTICE/SIM/XPS Level 3 and Level 4 data at the GSFC DAAC
- Archive of OMI Level 3 and Level 4 data at the GSFC DAAC
- Archive of HIRDLS, TES, MLS Level 1, Level 2 and Level 3 data at the GSFC DAAC
- Archive of products previously processed and archived
- Archive of Science Software I&T packages

3.1.1.3 Search and Order Capabilities

The ECS provides the following capabilities for search and ordering of data from the archive:

- Machine-to-Machine Gateway support between SIPS and ECS for data orders
- Directory and inventory search, including a user browse capability via the Version Zero (V0) System user interface
- Provide access to non-science data collections by a limited number of attributes and values
- V0 Gateway support for client requests for searches based on full ECS core metadata and product specific attributes and data subsetting for a limited number of data types
- Tracking order processing status via the System Management Subsystem (MSS)
- Configurable parameters to control the number of granules returned from a single search request
- Handling of variations on search areas and product-specific spatial representations
- Managing of orders via the V0 Gateway from the EOS Data Gateway (EDG), ECS Clearing House (ECHO), Ground Data System (GDS) and the orders submitted by the Spatial Subscription Server (SSS)
- The SSS provides an operator the interface to place standing orders (subscriptions) based on an ECS event and manage subscription status

- The Data Pool provides an operator the interface to manage insert processes, queues, collection groups and collection themes for ECS and non-ECS collections

3.1.1.4 Data Distribution Capabilities

The ECS provides the following Data Distribution capabilities for users:

- Support writing files to CD-ROMs and Digital Linear Tape drives for distribution
- Support File Transfer Protocol (FTP) Push or Pull Subscriptions for users
- Support distributing science data products via FTP, 8mm tape, CD-ROM, DVD, and DLT. (**Note:** physical media may not be available through all ordering applications.)

3.1.1.5 Data Processing Capabilities

The ECS provides the following capabilities for user/operator data processing options:

- Support the regeneration and archive of products previously produced and archived
- Provide capability for operator deletion of granules
- Allow users to request a data processing request associated with a DAR at the time of submittal
- Support Quality Assurance (QA) processing of Terra (AM-1) science data products
- Provide operator interfaces for production and resource planning
- Support submission of Data Acquisition Requests (DARs) for data collection by the ASTER instrument aboard the Terra (AM-1) spacecraft
- Support to submit on-demand requests to the ASTER GDS for the generation of Level 1B products
- Support for five new production rules required for Aqua (PM-1)
- Automated support for on-demand requests for ASTER processing
- Support of production rules to allow multiple Level 0 granules as input into a single PGE
- Support of production rules for most recent granule and optional DPRs
- Processing of orbit, attitude, and ephemeris data into toolkit native format and HDF
- Production rules for the closest granule, the spatial pad, and orbit processing of run time parameters
- Provide capability to associate the ASTER browse granule for the L1A product with the ASTER L1B and DEM products

3.1.1.6 System Operation and Architecture

The ECS provides the following capabilities to support the system operations and processing architecture used to provide data and services for users:

- Provide capability for operator deletion of granules, their associated metadata and browse files
- Provide the associated communications network interfaces with the SCFs
- Provide operator interfaces for production and resource planning
- Support interfacing with the Advanced Space-borne Thermal Emission and Reflection Radiometer (ASTER) Ground Data System (GDS) for the submission of Data Acquisition Requests (DARs) for data collection by the ASTER instrument aboard the Terra (AM-1) spacecraft
- Support managing the startup and shutdown of system network components, user registration and profile administration, database and archive administration, system data and file back-up and restores, system performance tuning and resource usage monitoring, and other routine operator duties
- Support of DAR submittals and DAR query status between the ECS Java DAR Tool and the ASTER GDS via the MOJO Gateway
- Support the display of browse data as a result of a single user request from the search results screen
- Operations support to update certain ESDT attributes without requiring the deletion of the data collection
- Provide ESDTs to support MODIS, AIRS, and AMSR on Aqua (PM-1)
- Provide ESDTs to support TIM, SOLSTICE, SIM, XPS on SORCE
- Provide spatial container changes for ASTER L1A/L1B from bounded rectangle to gpolygon
- Provide Science Software Integration and Test (SSI&T) support for Aqua (PM-1) (AIRS and MODIS)
- Provide the capability for editing of ECS core attribute values
- Support the consolidation of trouble tickets using the Remedy Tool
- Provide fault recovery for mode management
- Provide the capability for startup and shutdown of an entire mode
- Provide the capability for the deletion of science data from the archive
- Provide the capability for the installation of ESDTs to insert and acquire archived data without the archive storage directory names

- Provide the capability for the persistence of asynchronous acquire requests, which do not have callback functions
- Provide for the storage of event information into the SDSRV database instead of flat files
- Provide the capability for the monitoring of the usage of memory by the Science Data Server
- Provide COTS packages to allow operations to generate customized reports from ECS databases
- Provide a single configuration registry database to replace the numerous ECS application configuration files
- Provide for the insertion of ECS and non-ECS granules into the Data Pool

3.1.1.7 Security

The ECS provides the following capabilities for system security:

- Encryption of the subscription server FTP password in the Science Data Server database
- User authorization checks to restrict data set access at the granule level based on data quality information
- SDP Toolkit support for thread safe concurrent processing by the science software
- Secure Transfer of data files from Data Providers upon request
- System data and file backups and restores

3.1.1.8 DAAC/External System Support

ECS Release 7.11 has been distributed to five site locations including:

1. The System Management Center (SMC), located at the Goddard Space Flight Center (GSFC),
2. The DAAC at Goddard Space Flight Center (GSFC),
3. The DAAC at the Langley Research Center (LaRC),
4. The DAAC at Land Processing (LP), and
5. The DAAC at the National Snow and Ice Center (NSIDC)

The ECS Release 7.11 communications network includes the National Aeronautics and Space Administration (NASA) and the NASA Integrated Services Network (NISN). These portions of the network are physically located at the SMC and at the DAAC sites. The communications network connects ECS to data providers at the EDOS, NOAA Affiliated Data Center (ADC), and the EOSDIS Version 0 system.

The data users for Release 7.11 are the science user community connected to the four DAACs, the SCFs, and the ASTER GDS.

1. SMC Support:

   - SMC capabilities include overall ECS system performance monitoring, coordinating, and setting system-wide policies and priorities

2. GSFC Support:

   - AIRS/HSB/AMSU instrument data including the receipt of AIRS/HSB/AMSU Level 0 data and the GSFC archive, production, and distribution of Levels 1, 2 and 3 data and data products

   - ECS Release 7.11 provides communications network interfaces to support the Data Assimilation Office (DAO) in the receipt, archive, and distribution of data from NOAA's National Centers for Environmental Prediction (NCEP) and Terra instruments. This includes the ingest of NCEP Levels 1-3 data sets and the archive, production, and distribution of ECS Level 4 data by GSFC

   - The HIRDLS SIPS regularly receives HIRDLS Level 0 data from the GSFC DAAC via an ECS subscription. The Level 0 data is processed into Level 1, 2 and 3 data at the HIRDLS SIPS, and then transferred to the GSFC DAAC via the ECS-SIPS interface. The GSFC DAAC archives all HIRDLS standard data products and provides distribution services for the data

   - The MLS SIPS regularly receives MLS Level 0 data from the GSFC DAAC via an ECS subscription. The Level 0 data is processed into Level 1, 2 and 3 data at the MLS SIPS, and then transferred to the GSFC DAAC via the ECS-SIPS interface. The GSFC DAAC archives all MLS standard data products and provides distribution services for the data

   - OSIPS receives OMI Level 0 data from the GSFC DAAC via an ECS subscription. The Level 0 data is processed into Level 1 and 2 data at the OSIPS. The OSIPS transfers the Level 1 and 2 data to the GSFC DAAC for archival and distribution

   - TIM/SOLSTICE/SIM/XPS instrument data including the receipt of TIM/SOLSTICE/SIM/XPS Level 0 data and the GSFC archive, and distribution of Level 3 and Level 4 data and data products

   - GSFC DAAC capabilities include:

     - Ingest of MODIS, AIRS, HSB and AMSU Level 0 data and related ancillary data

     - Receipt of higher-level MODIS atmospheric and ocean products from MODAPS, via the SIPS interface, for archival and distribution

     - Production, archival, and distribution of the higher-level products for AIRS/HSB/AMSU data

     - Archival and distribution of the higher-level products for TIM/SOLSTICE/SIM/XPS data

3. LaRC Support:

   - ECS Release 7.11 provides a communications network and data/information management support for MISR instrument data including the receipt of MISR level 0 data and the LaRC archive, production, and distribution of levels 1, 2 and 3 data and data products
   - ECS Release 7.11 provides a communications network and data/information management support for MOPITT instrument data including the receipt of MOPITT level 0 data, the LaRC archive, and distribution of levels 1, 2 and 3 data
   - ECS Release 7.11 provides a communications network and data/information management support for TES instrument data including the receipt of TES level 0 data and the LaRC archive, and distribution of levels 1, 2 and 3 data
   - LaRC DAAC capabilities include:
     - Ingest of MISR and MOPITT Level 0 and related ancillary data
     - Production, archival, and distribution of the higher-level products for MISR
     - Receipt of higher-level MOPITT products from the MOPITT SCF, via the SIPS interface, for archival and distribution
     - Receipt of SAGE III Level 0 from the SAGE III MOC and the distribution of this data to the SAGE III SCF for processing
     - Receipt of higher-level SAGE III products from the SCF, via the SIPS interface, for archival and distribution
     - Receipt of ACRIM products (Level 0, ancillary data and Level 2 results) from the SCFs, via the SIPS interface, for archival and distribution
     - Receipt of TES Level 1 and 2 data including algorithm and associated software packages, metadata, production histories, ancillary data and Quality Assessment (QA) data for archival and distribution

4. LP DAAC Support:

   - ECS Release 7.11 provides a communications network and data/information management support for ASTER instrument data including the receipt of ASTER level 1 data on magnetic tape or electronically at LP DAAC from Japan, and the production and distribution of higher level ASTER products by LP DAAC
   - LP DAAC capabilities include:
     - Ingest of ASTER Level 1A/1B, with ancillary data needed for production
     - Production, archival and distribution of ASTER products
     - Receipt of higher level MODIS land products from MODAPS, via the SIPS interface, for archival and distribution

5. NSIDC Support:

   - AMSR-E instrument data including the receipt of level 0 data from EDOS at ECS, and the NSIDC archive and distribution of levels 1, 2 and 3 data. The Level 1A data is received from the NSIDC V0 DAAC while the level 2 and 3 data is received from the AMSR-E SCF via the SIPS interface
   - ECS Release 7.11 supports the ingest of ICESat GLAS level 1, level 2, level 3 and ancillary input data for archive and distribution at the NSIDC DAAC using the standard SIPS interface. The ECS also archives GLAS level 0 data received from EDOS
   - NSIDC DAAC capabilities include:
     - Receipt of higher-level MODIS snow and ice products from MODAPS, via the SIPS interface, for archival and distribution
     - Ingest of AMSR-E Level 0 data and related ancillary data
     - Receipt of the AMSR-E higher-level products via the SIPS interface, for archival and distribution
     - Ingest of GLAS Level 0 data and related ancillary data
     - Distribution of GLAS Level 0 to the SCF for higher-level processing
     - Receipt of the GLAS higher level products from the SCF, via the SIPS interface, for archival and distribution

6. SCF Support:

   - During the initial period after launch, the MOPITT higher-level products are generated at the SCF and provided to the ECS via the SIPS interface
   - ECS Release 7.11 supports receiving SAGE III Level 0 data from the SAGE III MOC, provides the level 0 data to the SAGE III SCF, and receives higher level products from the SCF via the SIPS interface
   - ECS Release 7.11 supports receiving ACRIM L0 data and higher level products from the SCF via the SIPS interface

7. MODAPS Support

   - ECS Release 7.11 provides a communications network and data/information management support for MODIS instrument data including level 0 data: archive, production, and distribution of levels 1A and 1B including distribution of the 1A and 1B data to the MODIS Data Processing System (MODAPS) for higher-level processing. The receiving of higher-level MODIS products from MODAPS via the SIPS interface is also supported

3.2 Release 7.11 Architecture Overview

The ECS Release 7.11 architecture comprises the logical items listed here. Commercial Off The Shelf (COTS) software and hardware are used, to the extent possible, to implement the ECS functionality of these logical items.

- System
- Segments
- Subsystems
- Computer software configuration items (CSCIs)
- Computer software components (CSCs)
- Processes

ECS Release 7.11 was built of the following two segments.

- CSMS – Communications and Systems Management Segment
- SDPS – Science Data Processing Segment

Each segment was in turn built of the following subsystems:

- CSMS: CSS – Communications Subsystem
  ISS – Internetworking Subsystem
  MSS – System Management Subsystem
- SDPS: CLS – Client Subsystem
  DMS – Data Management Subsystem
  DPS – Data Processing Subsystem
  DPL – Data Pool Subsystem
  DSS – Data Server Subsystem
  INS – Ingest Subsystem
  OMS – Order Management Subsystem
  OWS – OGC Web Services
  PLS – Planning Subsystem
  SSS – Spatial Subscription Server Subsystem

Hierarchical Definitions

System: A stand-alone composite of hardware, facilities, material, software, services, and personnel required for operation based upon a defined set of system level requirements and designed as a related set of capabilities and procedures.

Segment: A logical and functional subset of related capabilities, implemented with COTS hardware and COTS and custom developed software to satisfy a defined subset of the system level requirements.

Subsystem: A logical subset of Segment related capabilities, implemented with COTS hardware and COTS and custom developed software to satisfy a defined subset of segment level requirements.

CSCI: A logical subset of Subsystem related capabilities, implemented with COTS and custom developed software to satisfy a defined subset of the subsystem level software requirements.

CSC: A logical subset of CSCI related capabilities, implemented with COTS and custom developed software to satisfy a defined subset of the CSCI level software requirements.

Process: A logical and functional set of software, written in a specific order and in a defined manageable size to manipulate data as part of a product-generating algorithm. A process is a separately compiled executable (i.e., binary image). A process can use infrastructure library calls, system service calls, COTS service calls, and application programming interfaces to manipulate data to generate products.

Figure 3.2-1 is a hierarchical software diagram. The hierarchical software diagram depicts an example of the decomposition levels used in the ECS design and described in this document. The diagram is also a graphical representation of the terms just described.

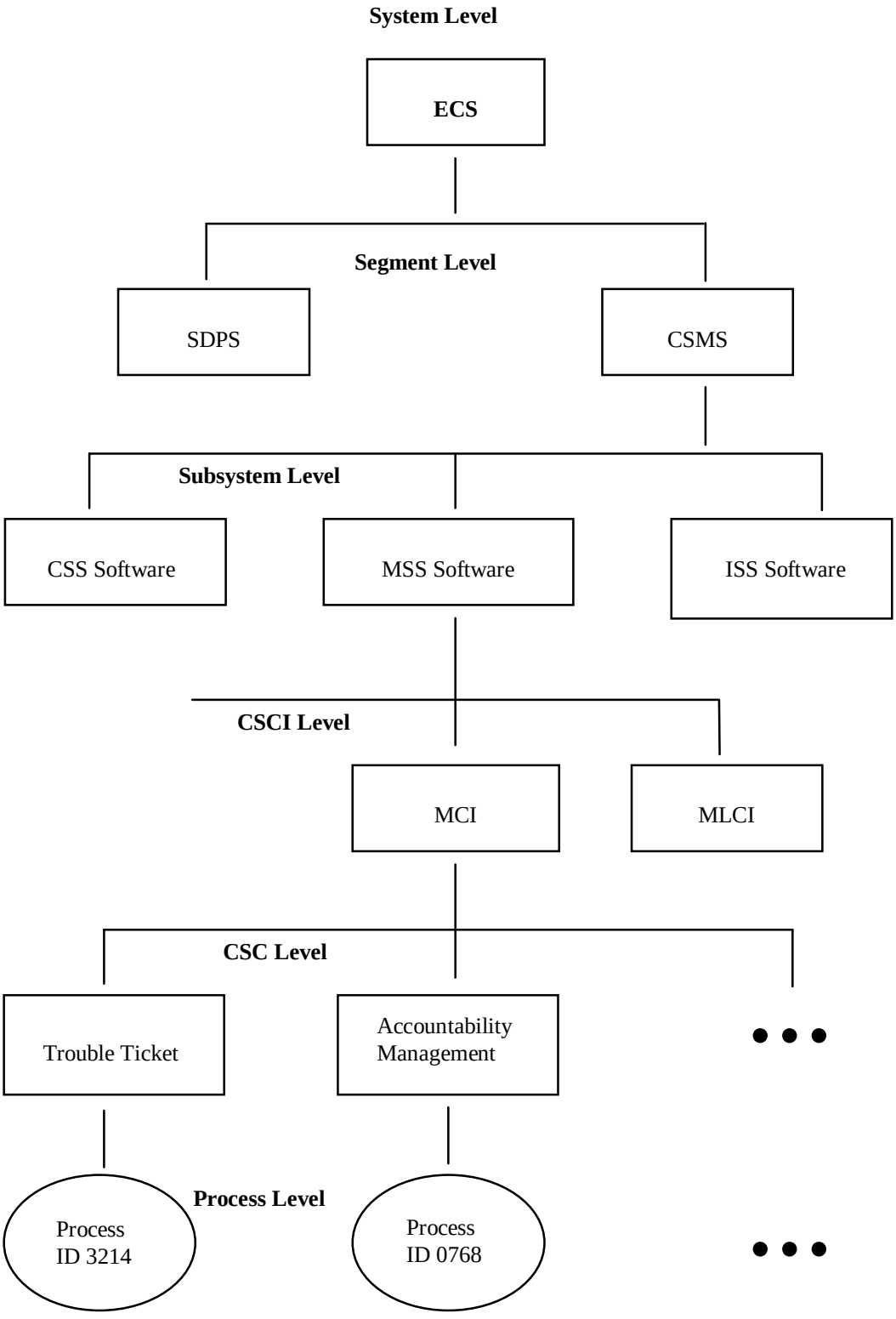


Figure 3.2-1. Example Hierarchical Software Diagram

3.2.1 Release 7.11 Context Description

ECS Release 7.11 provides the capability to collect and process satellite science data as depicted in Figure 3.2-2.

The Science Data Processing and Communications and Systems Management are the two segments of the ECS Release 7.11 described in this document. The Science Data Processing Segment (SDPS) provides science data ingest and production, search and access functions, data archive, and system management capabilities. The SDPS receives Terra (AM-1) and Aqua (PM-1) Level 0 science data from EDOS. The SDPS exchanges data with the ASTER GDS and other affiliated data centers to obtain science and other data (i.e., engineering and ancillary) required for data production. It also connects with the ASTER GDS to submit ASTER Data Acquisition Requests (DARs) for the collection of science data by the ASTER instrument. Science algorithms, provided by the Science Computing Facilities (SCFs), are used in data production to transform data into higher-level products (Level 1 to Level 4 products) for research. The ECS project uses SCF expertise to support the Quality Assurance activities of using the results of the Science Software Integration and Test (SSIT) activities to process data and verifies the data production science algorithms prior to actual data production. The Communications and Systems Management Segment (CSMS) provides the communications infrastructure for the ECS and systems management for all of the ECS hardware and software components. The CSMS provides the interconnection between users and service providers within the ECS, transfer of information between subsystems, CSCIs, CSCs, and processes of the ECS.

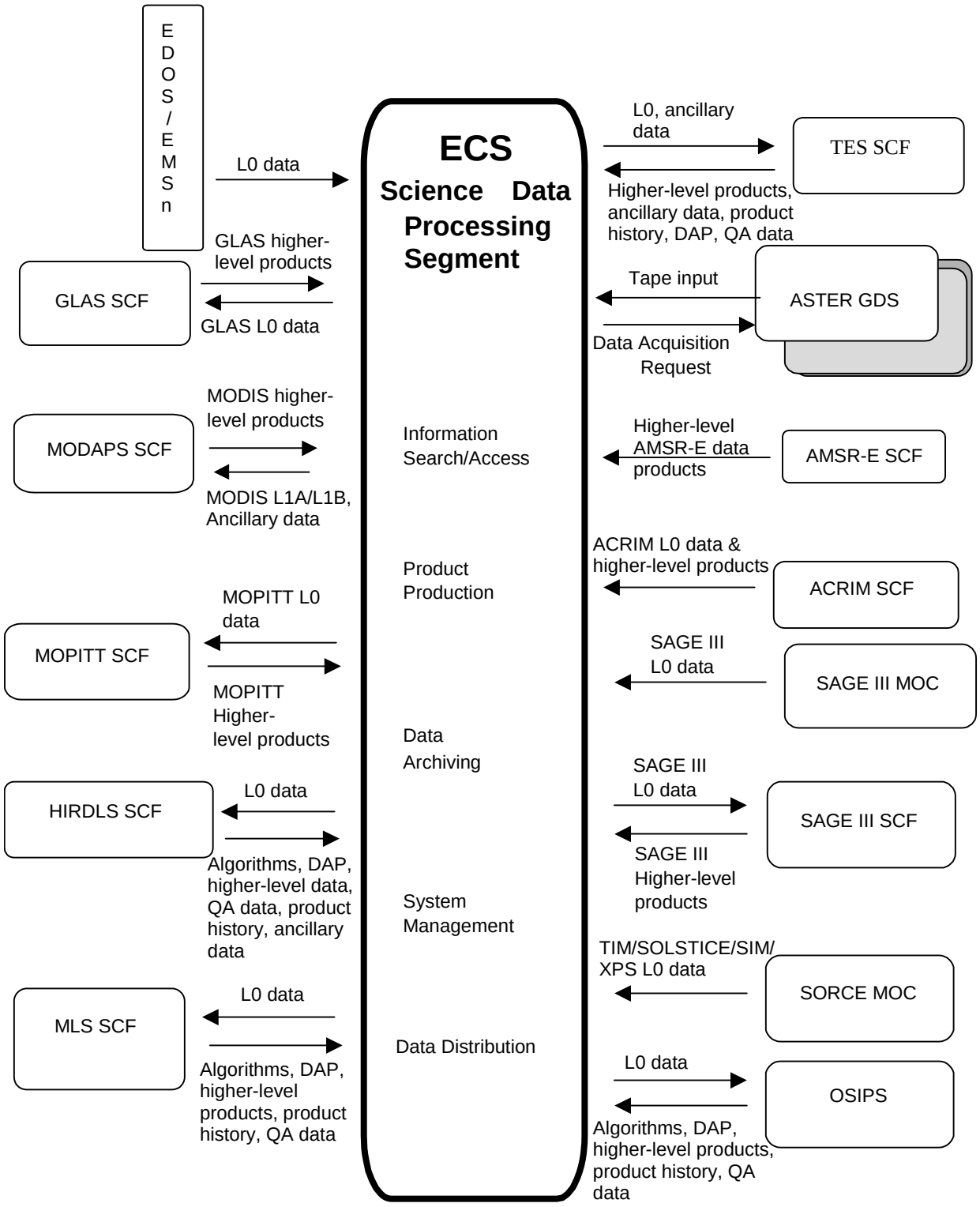


Figure 3.2-2. Release 7.11 Context Diagram

The remaining sections of this document provide an overview of the ECS Release 7.11 design and as such do not deal specifically with the configuration of components at each EOSDIS site. For more information on the site unique configurations, refer to the 920-series of General documents. Each of the segments consists of subsystems as specified in Section 3.2.

3.2.2 Release 7.11 Architecture

3.2.2.1 Subsystem Architecture

The ECS SDPS subsystems are depicted in Figure 3.2-3. A subsystem consists of the Commercial Off The Shelf (COTS) and/or ECS developed software and the COTS hardware needed for its execution. The SDPS subsystems can be grouped into a 'Push' or 'Pull' category of functionality with the exception of DSS. As shown in the subsystem architecture diagram, the information search and data retrieval makes up the 'Pull' side of the ECS architecture/design and consists of the CLS, DMS, OMS, SSS, DPL and also uses the DSS functionality described on the 'Push' side of the ECS architecture. Data capture (ingest of data), storage management, planning and data processing of satellite or previously archived data from other sites make up the 'Push' side of the ECS architecture/design and consists of the DSS, INS, PLS, and DPS. This document describes the software and hardware components of each subsystem. However, since the hardware configurations differ between the sites, the hardware descriptions in this document are at a generic level. Specific hardware and network configurations for each site are documented in the 920 and 921 series technical documents.

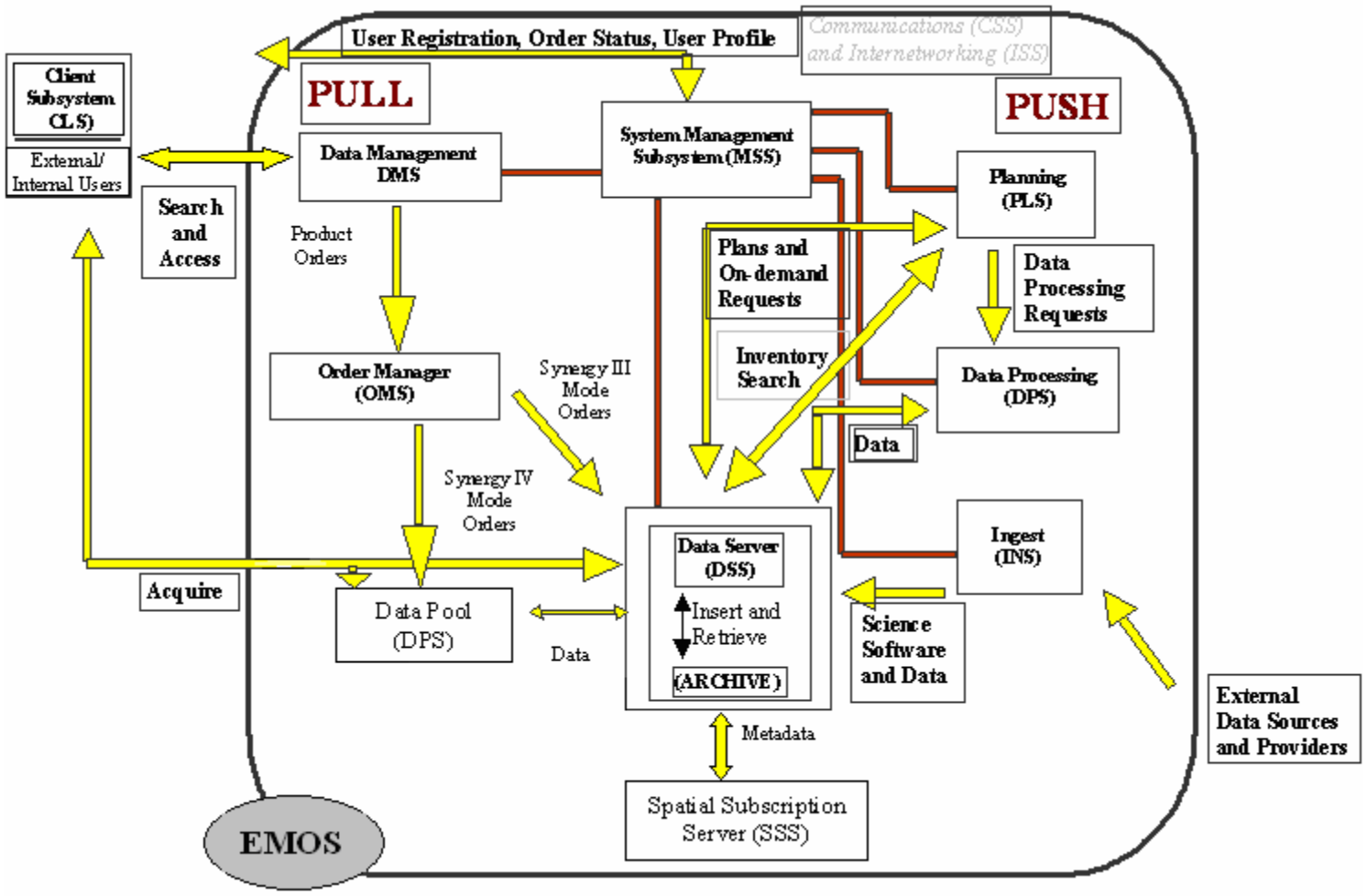


Figure 3.2-3. Subsystem Architecture Diagram

The ECS architecture/design and consists of:

1. The DSS with the functions needed to
   - Archive (insert) science data
   - Search for and retrieve archived data
   - Manage (create and delete items from) the archives
   - Stage (store on physical media temporarily) data resources needed as input to data processing or resulting as output from their processing execution
2. The INS with interfaces to external suppliers (such as EDOS) to:
   - Transfer data into SDPS and temporarily store the data
   - Provide staging capabilities for data waiting to be archived
   - Provide operator interfaces for managing ingest operations
3. The DPS and PLS with:
   - DPS dispatching and monitoring the execution of the science software, and interfaces to DSS to stage the input data needed and archive the generated data
   - PLS providing long and short term planning of science data processing of the production environment and production resources

Information search, data retrieval, and data distribution makes up the 'Pull' side of the ECS architecture/design and consists of:

1. The CLS, DMS, OMS, SSS and DPL with:
   - CLS providing user interfaces for data search and retrieval to science users and operators
   - DMS providing support for data search and retrieval across all ECS sites in conjunction with the DSS at each site. The DMS also provides a gateway as the interface to the Version 0 (V0) Information Management System (IMS) using the V0 IMS protocol
   - Other SDPS subsystems (i.e., CLS, PLS, DPS, DMS, and INS) use configuration files to determine the location of data and the services, offered by the DSS, for a particular type of data or distributed data that has been processed

     OMS managing all orders received from the DMS V0 Gateway (i.e., from EDG, ECHO and GDS users), the Machine-to-Machine Gateway, and the Spatial Subscription Server
   - SSS supporting the creation, viewing and updating of subscriptions and the creation, viewing and deletion of bundling orders (specification of distribution packages and criteria for package completion)

- DPL supporting the search, order, and distribution of selected granules with associated metadata and browse granules (if available)

CSMS – The following subsystems are the CSMS subsystems, which interact with and support the SDPS to complete the ECS architecture.

2. The MSS with:
   - Hardware and software baseline and configuration management
   - Trouble ticketing and nonconformance report (NCR) tracking
   - System start-up and shut-down
   - Fault and performance monitoring for networks, platforms, and software applications
   - User account management and user order tracking
3. The CSS with:
   - Control Center System (CCS) Middleware provides a common Name Server, which packages the common portions of the communication mechanisms into global objects to be used by all subsystems. The Name Server provides a set of standard CCS Proxy/Server classes, which encapsulates all of the common code for middleware communications (e.g., portals, couplers, RWCollectables, etc.)
   - Libraries with common software mechanisms for application error handling, aspects of recovering client/server communications; Universal References to distributed objects and interfaces to e-mail, file transfer and network file copy capabilities
   - External gateways to translate from the CSMS internal protocols, based on sockets, to protocols acceptable by external systems such as ASTER GDS and vice versa
4. The ISS with:
   - Networking hardware devices (e.g., routers, switches, hubs, cabling, etc.) and their respective embedded software. For more information on site unique configurations, refer to the 920-series of General documents

4. Subsystem Description

Design Description Organization and Approach

This section presents a subsystem-by-subsystem overview description of the "as-built" EMD. The current high-level design information is provided for the Hardware Configuration Items (HWCI), Computer Software Configuration Items (CSCI), and Computer Software Components (CSC) for each subsystem and is being delivered to the DAACs in drop increments.

The SDPS and CSMS subsystem descriptions include:

- Subsystem functional overviews with a subsystem context diagram and a table of interface event descriptions
- CSCI descriptions with a context diagram and a table with interface event descriptions
- Architecture Diagrams, Process Descriptions, and Process Interface Event Tables. The Architecture Diagrams show the processes of the CSCI/CSC and how these processes connect with other CSCIs and CSCs of the same subsystem and the interfaces with other subsystems and external entities such as Operations, External Data Providers and Users. These processes and the supporting libraries are listed in Appendices A (Software Processes) and B (Software Libraries)
- Data Store descriptions for each CSCI in each SDPS/CSMS subsystem. The Data Stores are identified with the software name and shown in the architecture diagrams either as single data stores or as a group of data stores with a generic name such as "Data Stores" or "database"
- Hardware descriptions of the subsystem hardware items and the fail-over strategy

The convention used for Context and Architecture diagrams includes using circular shapes to show the subject subsystems, CSCIs, CSCs, or processes (with name in bold), elliptical shapes to show associated CSCIs, CSCs, or processes within a given subsystem and squares or rectangles to show external subsystems, CSCIs, CSCs, and processes. Data stores are shown using the data store or database name with horizontal lines, one above and one below the name. An interface event is data, a message (which includes a notification or status); a command, request or status code passed between subsystems, CSCIs, CSCs, or processes. The convention used to identify events is a straight line between two objects labeled with a phrase beginning with an action-oriented word to best describe the event. The arrow on the event line indicates an origination point and to where the event is directed. A direct response to an event is not always shown on the diagram because sometimes there is no response (e.g., for an insert or delete request) and other times the response comes from another part of the EMD. Interface events are identified in the interface event or process interface tables starting with the interface event at the top or middle of the diagram and going clock-wise around the diagram. The external interface subsystem is identified in the interface event description and is in bold to assist with the location of the interface events on the diagram. If there are two items in bold, there are two different interfaces

(Subsystems, CSCIs, or CSCs) requesting the same interface event. These conventions are consistent with other EMD documentation. The convention for naming the EMD processes is Ec <subsystem abbreviation> meaningful name. The *Ec* identifies the process as an EMD developed process versus a Commercial Off The Shelf (COTS) product. The *subsystem abbreviations* are listed subsystem-by-subsystem.

- Cl for CLS
- Cs for CSS
- Dl for DPL
- Dm for DMS
- Dp for DPS
- Ds for DSS
- In for INS
- Ms for MSS
- Nb for SSS
- Om for OMS
- Pl for PLS
- Ow for OWS

The *meaningful name* identifies the process and its functionality within the subsystem, CSCI, or CSC. An example is EcDsScienceDataServer, which identifies an EMD-developed DSS process called the Science Data Server. Some names within an architecture diagram do not follow this convention because the names are COTS product names. All COTS product names are kept for simplicity and to adhere to licensing and trademark agreements. The remaining names that do not follow the naming convention are imbedded throughout the system and would require time to replace and cause operational disruptions. These names will be cleaned up during the final maintenance stages of the contract if directed by the customer.

Object-oriented modeling and design

Object-oriented modeling and design is a new way of thinking about problems using models organized around real-world concepts. The fundamental construct is the object, which combines both data structure and behavior in a single entity. Objected-oriented models are useful for understanding problems, communicating with application experts, modeling enterprises, preparing documentation and designing programs and databases.[1]

Superficially the term "object-oriented" means that we organize software as a collection of discrete objects that incorporate both data structure and behavior. This is in contrast to conventional programming in which data structure and behavior are only loosely connected. There is some dispute about exactly what characteristics are required by an object-oriented approach, but generally include four aspects: identity, classification, polymorphism and

inheritance.[1][1] *Identity* means that data is quantized into discrete, distinguishable entities called *objects*. A paragraph in my document, a window on my workstation and a white queen in a chess game are examples of objects. Objects can be concrete, such as a file, or conceptual, such as a *scheduling policy* in a multi-processing operating system. Each object has its own inherent identity. In other words, two objects are distinct even if all their attribute values (such as name and size) are identical.[1]

In the real world an object simply exists, but within a programming language each object has a unique *handle* by, which it can be uniquely referenced. The handle may be implemented in various ways, such as an address, array index or unique value of an attribute. Object references are uniform and independent of the contents of the objects, permitting mixed collections of objects to be created, such as a file system directory that contains both files and sub-directories.[1]

Classification means that objects with the same data structure (attributes) and behavior (operations) are grouped into a *class*. Paragraph, Window, and ChessPiece are examples of classes. A *class* is an abstraction that describes properties important to an application and ignores the rest. Any choice of classes is arbitrary and depends on the application.[1]

Each class describes a possibly infinite set of individual objects. Each object is said to be an instance of its class. Each instance of the class has its own value for each attribute but shares the attribute names and operations with other instances of the class. An object contains an implicit reference to its own class: it "knows what kind of a thing it is."[1]

Polymorphism means that the same operation may behave differently on different classes. The *move* operation, for example, may behave differently on the *Window* and *ChessPiece* classes. An *operation* is an action or transformation that an object performs or is subject to. *Right justify, display* and *move* are examples of operations. A specific implementation of an operation by a certain class is called a *method*. Because an object-oriented operator is polymorphic, it may have more than one method implementing it.[1][2]

In the real world, an operation is simply an abstraction of analogous behavior across different kinds of objects. Each object "knows how" to perform its own operations. In an object-oriented programming language, however, the language automatically selects the correct method to implement an operation based on the name of the operation and the class of the object being operated on. The user of an operation need not be aware of how many methods exist to implement a given polymorphic operation. New classes can be added without changing existing code, provided methods are provided for each applicable operation on the new classes.[1]

Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship. A class can be defined broadly and then refined into successively finer *subclasses.* Each sub-class incorporates, or *inherits* all the properties of its *super-class* and adds its own unique properties. The properties of the superclass need not be repeated. For example,

---

[1] Object-oriented Modeling and design, James Rumbaugh et all, copyright 1991 by Prentice-Hall, Inc. ISBN 0-13-629841-9

[2] Object-oriented Modeling and design, James Rumbaugh et all, copyright 1991 by Prentice-Hall, Inc. ISBN 0-13-629841-9

ScrollingWindow and *FixedWindow* are subclasses of *Window*. Both subclasses inherit the properties of *Window*, such as a visible region on the screen.[1]

The EMD is a large, complex data storage and retrieval system used to store and retrieve large amounts of science and science-related data. The system was designed using an object oriented design approach. With so many objects and the sizes of some of them, it is necessary to have some insight into the amount of memory being utilized within the EMD. The information about to be presented is a brief look at the memory management of the "key" (top ten utilized) objects within the EMD subsystems.

In this object oriented system design, objects are created and used via classes throughout the system to help perform the functions and meet the needs of the system. The objects for the EMD are very numerous, sometimes very large and cannot be provided in their entirety at this time. However, presented in the table below are the "key" objects for this system and how they are created, passed and deleted within the EMD.

Introduction to memory management approaches and memory usage table

Good memory management in some applications is both important and requires significant planning and development time. Many important EMD applications are large, long running, multi-threaded, heavy memory users and therefore are prime candidates for improved memory management.[1]

Improper memory management can result in memory leaks, fast memory usage growth or large application footprints and random crashes. EMD servers are periodically purified for memory leaks and there is a history of progress in this area. Similar work should be expected to continue as development and maintenance continues.

Long running server like applications that are free from memory leaks can nonetheless have significant memory and Central Processing Unit (CPU) usage performance degradation. A common culprit is heap fragmentation. The repeated allocation and deallocation of memory (such as with the new and delete operators of C++) can result in a large number of unusable free blocks of memory. They are free blocks but are interspersed with non-free blocks. They become unusable since they are not contiguous (fragmented) and as time goes by, it becomes harder and harder for the OS to service requests for more memory. Such situations even lead to crashes of other, non-offending applications running in the same box.

There are strategies, tools and software to avoid both memory leaks and fragmentation. This includes but is not limited to:

- Periodic application of purification software (already an EMD practice)
- Software design, which uses dynamic memory as little as possible, such as automatic storage or COTS data structures

[1] Object-oriented Modeling and design, James Rumbaugh et all, copyright 1991 by Prentice-Hall, Inc. ISBN 0-13-629841-9

- Class-level memory management to allocate large chunks of memory instead of one class instance at a time ("Effective C++" by Scott Meyers and "Advanced C++" by James Coplien address this technique)
- Non-class level memory pools and
- COTS heap manager

Table 4-1 below is provided in case further memory management improvements are desired. Given operator or field input of seemingly inefficient memory or CPU usage, this table can be used to help target specific EMD subsystems, servers and frameworks or classes for improvement. It can be decided to apply some of the approaches at one level (e.g., on one guinea pig server or class) or perhaps experiment with changing the entire EMD C++ system with the use of a COTS heap manager. In any case, a great deal of planning and manpower is required.

Table 4-1. Memory Management Table (1 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
DSS - SDSRV	EcDsScienceDataServer	DsSrRequest	Executes requests based on request type. Base class for DsSr<funct>Request where <funct> = Add, Acquire, ESDT, Insert, Search, and Generic	EcDsScienceDataServer (class: DsSrConnectionMaker)	Not passed. Request is immediately executed in DsSrConnectionMaker	DsSrConnection Maker	One per DsSrRequest	Object is deleted when the DsSrRequest is completed.
	EcDsScienceDataServer	DsSrWarmStartManager	Singleton class controlling processing of asynchronous requests.	EcDsScienceDataServer (class: DsSrManagedServer)	not passed	EcDsScienceDataServer	Either 1 instance or no instances for all of SDSRV.	Static singleton class. Object is deleted when the Science Data Server goes down.
	Clients, EcDsScienceDataServer	DsShSRequestReal	This class provides a server interface to the server's request distributed object. It inherits from the DCE-generated server request class, and adds functions to provide stronger type checking.	Clients, EcDsScienceDataServer	EcDsScience DataServer or other servers related	When request is finished or server goes down	1 per client request.	This class communicates between clients and EcDsScienceDataServer.

Table 4-1. Memory Management Table (2 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	Clients and EcDsScienceDataServer	GlParameterList	This library is used by many subsystems to provide a general-purpose list object for storing various scalar and complex data types.	Clients and EcDsScienceDataServer	EcDsScienceDataServer and EcDsDistributionServer	Clients	Could have many in each request	To group one or more GlParameter derived classes that store the various parameter types required building commands. Any Gl type, including embedded GlParameterLists can be inserted into a GlParameterList, making it recursive in design.
	EcDsScienceDataServer	DsGeESDT	Inherit from the public class - provides basic ESDT functionality.	EcDsScienceDataServer	EcDsDistributionServer, EcDsStRequestManagerServer and other related applications	EcDsScienceDataServer and other related applications	1 per granule	This class provides functionality common to all SDSRV data types.
	DsSrManagedServer::DsSrStart()	DsBtSbsrvNotifier	This class is used to facilitate communication between the SDSRV and SBSRV through an event queue.	EcDsScienceDataServer [class::method = DsSrManagedServer::DsSrStart()]	not passed	EcDsScienceDataServer [class::method = DsSrManagedServer::DsSrStart()]	Configurable	Only one instance per session is created. The object goes away when the server goes down.

Table 4-1. Memory Management Table (3 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDsScienceDataServer [function = DsGeESDT::Insert()]	DsMdMetadata	This class is used as a container class for metadata.	EcDsScienceDataServer [function = DsGeESDT::Insert()]	not passed	See comments/remarks column.	1 per DsGeESDT	When this object is instantiated, it uses the local memory manager. The object can be saved to the database if the user is executing an insert.
	EcDsScienceDataServer [class:::method= DsSrGenCatalogPool:::DsSrGenCatalogPool()]	DsMdCatalog	This class is used to manage catalog pools.	DsSrGenCatalogPool:::DsSrGenCatalogPool()	not passed	EcDsScienceDataServer	Depends on the configured pool size.	There are three default pools for catalogs: SEARCH, INSERT and DEFAULT. The object goes away when the server goes down.
	EcDsScienceDataServer	DsDbInterface	Database (Sybase) interface class to encapsulate database related services such as: connect, execute, fetch result.	EcDsScienceDataServer	not passed	When EcDsScienceDataServer is down	2 per DsMsCatalog	User can connect to DB, execute SQL statements, verify connection states and disconnect from the database.

Table 4-1. Memory Management Table (4 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	Clients	DsClESDTReferenceCollector	Provides the primary interaction mechanism for client software.	Clients	EcDsScienceDataServer	EcDsSdsrvTestor EcDsTsClientDriver or when clients go down	1 per client connection	As its name implies, it is a collector of the DsClESDTReference object referred to as the clients "working collection", which is populated with the results of service requests such as "Acquire", "Insert", "Search."
DSS - STMGT	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer	DsStDictionary	Maps a string name to a string value.	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveSever EcDsStFtpServer	not passed not passed not passed not passed not passed not passed not passed	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer	Used in Singleton DsStStProcTable	Object is deleted when the server goes down.

Table 4-1. Memory Management Table (5 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
DSS - STMGT	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer	DsStStProcTable	A singleton used to create a dictionary that associates a stored procedure name with a list of parameters in the order that they appear in the stored procedure declaration.	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer	not passed not passed not passed not passed not passed not passed not passed	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer	Singleton	Object is deleted when the server goes down.

Table 4-1. Memory Management Table (6 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer EcDsDistributionServer EcInReqMgr EcInGran	DsStRemoteTransaction	Handles the remote transactions for stored procedure call. It converts the request in a format, which can be passed across a network interface.	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer EcDsDistributionServer EcInReqMgr EcInGran	not passed not passed not passed not passed not passed not passed not passed not passed not passed	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer EcDsDistributionServer EcInReqMgr EcInGran	4 per request (client request and result, server request and result)	

Table 4-1. Memory Management Table (7 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer EcDsDistributionServer EcInReqMgr EcInGran	DsStRemoteSP	Parent class of DsStRemoteTransaction.	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer EcDsDistributionServer EcInReqMgr EcInGran	not passed not passed not passed not passed not passed not passed not passed not passed not passed	EcDsStRequestManagerServer EcDsStCacheManagerServer EcDsStStagingDiskServer EcDsStArchiveServer EcDsStFtpServer EcDsDistributionServer EcInReqMgr EcInGran	Multiple times per request	
	DsStRequestManagerServer	DsStRequest	Describes all possible states of a request.	DsStRequestManagerServer	not passed	DsStRequestManagerServer	1 per request	
	DsStRequestManagerServer	DsStRequestQueue	Provides queuing mechanism for DsStRequest objects.	DsStRequestManagerServer	not passed	DsStRequestManagerServer	Singleton	

Table 4-1. Memory Management Table (8 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDsScienceDataServer(DLL), EcDsDistributionServer, EcDsStArchiveServer	DsStFileParameters	Data structure to maintain the file related parameters.	EcDsScienceDataServer (DLL), EcDsDistributionServer, EcDsStArchiveServer	not passed not passed not passed	EcDsScienceDataServer (DLL), EcDsDistributionServer, EcDsStArchiveServer	Multiples times per request within Archive server and Archive client	
DSS - STMGT	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	DsStCopyService	File I/O operation and checksuming for copy service..	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	not passed not passed not passed	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	One per CacheManagerServiceThread, One per StagingDiskServiceThread, One per ArchiveWriteThread	The item should be the DsStCopyService class rather than the DsStCopyService::Copy function.

Table 4-1. Memory Management Table (9 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	DsStFtpService::FtpStor()	File I/O operation for Copy service.	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	not passed not passed not passed	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	One per CacheManagerServiceThread, One per StagingDiskServiceThread, One per ArchiveWriteThread	
	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	DsStFtpService::FtpRetr()	File I/O operation for Copy service.	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	not passed not passed not passed	EcDsStCacheManagerServer, EcDsStStagingDiskServer, DsStArchiveServer	One per CacheManagerServiceThread, One per StagingDiskServiceThread, One per ArchiveWriteThread	
DSS - DDIST	EcDsDistributionServer	DsDdMedia	Contains media drivers and request level media information, like media type.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once per DDIST request.	

Table 4-1. Memory Management Table (10 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDsDistributionServer	DsDdDistRequestS	Contains request level information, like State and orderID.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once per DDIST request.	
	EcDsDistributionServer	DsDdDistListS	Contains pointers to granule and file information for the request.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once per DDIST request.	
	EcDsDistributionServer	DsDdMediaDist	Contains packaging information and has a one-to-one correspondence to physical media for the request. Note: requests can be sufficiently large to span more than one physical media.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created at least once per each DsDdMedia	
	EcDsDistributionServer	DsDdGranuleS	Contains granuleUR information.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created for each granule for each request.	
	EcDsDistributionServer	DsDdDistFileS	Contains file information, like file name & file size.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created for each file (archive data file, metadata file, etc.) in each granule	

Table 4-1. Memory Management Table (11 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	Created in most functions and most object constructors	RWCString	The rogue-wave string class used through all classes and functions.	Created in most functions and most object constructors.	An extensive list of functions.	An extensive list of functions.	Created in most functions and most object constructors.	
	Created in most functions.	DsDdLog	Utility for simplifying logging.	Created in most functions.	EcDsDistributionServer	EcDsDistributionServer	Created in most functions.	
	EcDsDistributionServer	DsDdActiveQueue	A Queue of requests that are in Active, Staging, and Transferring states.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once (singleton class)	
	EcDsDistributionServer	DsDdBaseQueue	Parent class of the remaining queues.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once (singleton class)	
	EcDsDistributionServer	DsDdConfiguration	Utility class for reading configuration information.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once (singleton class)	
	EcDsDistributionServer	DsDdDoneQueue	A Queue of requests in the Shipped, Failed or Cancelled states.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once (singleton class)	
	EcDsDistributionServer	DsDdHoldQueue	A Queue of requests that are in one of the suspended states.	EcDsDistributionServer	not passed	EcDsDistributionServer	Created once (singleton class)	

Table 4-1. Memory Management Table (12 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDsDistributionServer	DsDdPriorityQueue	A Priority Queue of requests that are pending. Uses DsDdThreadPool class in the process of selecting the next request for processing.	EcDsDistributionServer	not passed	EcDsDistribution Server	Created once (singleton class)	
	EcDsDistributionServer	DsDdReadyToShipQueue	A Queue of hard media requests waiting on the operator for selection to enter the Shipped state.	EcDsDistributionServer	not passed	EcDsDistribution Server	Created once (singleton class)	
	EcDsDistributionServer	DsDdRequestListS	List of all DDIST requests.	EcDsDistributionServer	not passed	EcDsDistribution Server	Created once (singleton class)	
	EcDsDistributionServer	DsDdScheduler	Class, which wakes up worker threads and assigns them to requests as selected by the DsDdPriorityQueue.	EcDsDistributionServer	not passed	EcDsDistribution Server	Created once (singleton class)	
	EcDsDistributionServer	DsDdThreadPool	Uses thread pools DB tables and procedures to select next request for processing, given request information and constraints.	EcDsDistributionServer	not passed	EcDsDistribution Server	Created once per each thread pool name (usually 5 - 50)	

Table 4-1. Memory Management Table (13 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
DMS	EcDmV0ToEcsGateway	GlParameterList::DeepAssign( )	Generic copy method for ECS composite class.	Used as part of query transport to SDSRV – therefore permeates through all request classes.	SDSRV (CSCI), PLANG (CSCI), Registry (CSC)	Where appropriate	Many	DMS code compensates for native leak by client side action
	EcDmV0ToEcsGateway	DmGwInventoryRequest	Handler for V0 Inventory Searches.	DmGwV0EcsRequestReciever	not passed	DmGwV0EcsRequestReciever	1 per inventory request	No significant leaks
	EcDmV0ToEcsGateway	DmGwBrowseRequest	Handler for V0 Browse Requests.	DmGwV0EcsRequestReciever	not passed	DmGwV0EcsRequestReciever	1 per browse request	No significant leaks, but for integrated browse may hold large amount of memory while active
	EcDmV0ToEcsGateway	DmGwProductRequest	Handler for V0 Product Requests.	DmGwV0EcsRequestReciever	not passed	DmGwV0EcsRequestReciever	1 per product request	No significant leaks
	EcDmV0ToEcsGateway	DmGwSpecializedCriteria	Representation of V0 SPECIALIZED_CRITERIA element.	EcDmV0ToEcsGateway	not passed	Where appropriate	Many per granule	Recursive class, potential for large memory usage. No significant leaks.
	EcDmV0ToEcsGateway	DmGwDirectoryRequest	Handler for V0 Directory Searches.	DmGwV0EcsRequestReciever	not passed	DmGwV0EcsRequestReciever	1 per directory request	No significant leaks
	EcDmV0ToEcsGateway	DmGwGranuleLevelSearch	SDSRV search client-side search object.	DmGwInventoryRequest	not passed	DmGwInventoryRequest	1 per dataset searched	No significant leaks
	EcDmV0ToEcsGateway	DmGwRequestReceiver	Listener / Dispatch class for EcDmV0ToEcsGateway.	DmGwManagedServer	not passed	DmGwManagerdServer	1 per process	No significant leaks

Table 4-1. Memory Management Table (14 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDmV0ToEcsGateway	EcRgDistOptions	Client class for distribution options provision.	DmGwDataset	not passed	DmGwDataset	1 per search	No significant leaks
	EcDmV0ToEcsGateway	EcRgSubsetOptions	Client class for subset options provision.	DmGwDataset	not passed	DmGwDataset	1 per search	No significant leaks
DPS	EcDpPrEM EcDpPrDeletion EcPlPREditor_IF EcPlOdMgr	DpPrDSSInterface	Interface to SDSRV.	EcDpPrEM EcDpPrDeletion EcPlPREditor_IF EcPlOdMgr	not passed	EcDpPrEM EcDpPrDeletion EcPlPREditor_IF EcPlOdMgr	# granules x DPR # interim files x DPR # input granules x DPR # input granules x OD DPR	
	EcDpPrEM	DpPrDataManager	Manages acquires and inserts of granules from/to SDSRV.	EcDpPrEM	not passed	EcDpPrEM	# granules x DPR	
	EcDpPrEM	DpPrExecutionManager	Supervisory program for DPR execution.	EcDpPrEM	not passed	EcDpPrEM	2 x DPR	
	EcDpPrEM	DpPrResourceManager	Manages disk allocation for files and CPUs for DPRs.	EcDpPrEM	not passed	EcDpPrEM	(# files + 2) x DPR	

Table 4-1. Memory Management Table (15 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDpPrJobMgmt EcDpPrEM	DpPrScheduler	Manages DPR construction in AutoSys and scheduling on computers.	EcDpPrJobMgmt	not passed	EcDpPrJobMgmt	1 x DPR and until server is brought down	
	EcDpPrEM	DpPrPcf	Constructs process control file.	EcDpPrEM	not passed	EcDpPrEM	1x DPR	
	EcDpPrJobMgmt	DpPrCotsManager	Interface to AutoSys.	EcDpPrJobMgmt	not passed	EcDpPrJobMgmt	2 x DPR	
	EcDpPrJobMgmt	DpPrJIL	Interface to AutoSys.	EcDpPrJobMgmt	not passed	EcDpPrJobMgmt	1 x DPR	
	EcDpPrEM	DpPrPge	Manages acquire of PGE.	EcDpPrEM	not passed	EcDpPrEM	1 x DPR	
	EcDpPrEM	DpPrFile	Helper class for DprPrDataManagement.	EcDpPrEM	not passed	EcDpPrEM	# files x DPR	
	EcDpPrDeletion	DpDeletionServer	Removes PDPS files that are no longer used.	EcDpPrDeletion	not passed	EcDpPrDeletion	until server is brought down	
	EcDpPrDeletionClient	DpDeletionProxy	Responsible for identifying files to be deleted.	EcDpPrDeletionClient	not passed	EcDpPrDeletionClient	2 x day	
PLS	EcPlPREditor	PlDpr	Data Processing Request class.	EcPlPREditor	not passed	EcPlPREditor	1 per DPR	Created as a member of a static pool which is deleted when PREditor is brought down
	EcPlPREditor	PlUserParameters	PGE Processing Parameters class.	EcPlPREditor	not passed	EcPlPREditor	Multiple per PGE	

Table 4-1. Memory Management Table (16 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcPlPREditor, EcPlOdMgr, EcPlSubMgr	PlDprData	Record that relates each input/output granule with a Data Processing Request.	EcPlPREditor, EcPlOdMgr, EcPlSubMgr	not passed not passed not passed	EcPlPREditor, EcPlOdMgr, EcPlSubMgr	1 per unavailable primary/alternate input	
	EcPlPREditor, EcPlOdMgr	PlDataGranule	Record for each input/output granule.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per input/output granule	
	EcPlPREditor, EcPlOdMgr	PlPge	Record of PDPS PGE information.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per Pge	Created as a member of a static pool which is deleted when PREditor or OdMgr are brought down
	EcPlOdMgr	PlHighLevelOrder	Record of ASTER OnDemand High Level Order.	EcPlOdMgr	not passed	EcPlOdMgr	1 per High Level order	Deleted after each ASTER OnDemand order is processed
	EcPlPREditor, EcPlOdMgr	PlDataScheduled	Record of Data Scheduled type PGE.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per Data Scheduled PGE	Deleted after PGE information is collected
	EcPlPREditor, EcPlOdMgr	PlDataTypeReq	Record of ESDT of Data Processing Request input.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per DPR input ESDT	
	EcPlPREditor, EcPlOdMgr	PlMetadataChecks	Record of required metadata checks for PGE inputs.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per PGE input if metadata checks are required	Deleted when the collection is destroyed

Table 4-1. Memory Management Table (17 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcPlPREditor	PlUserParameters	Record of user defined processing parameter values.	EcPlPREditor	not passed	EcPlPREditor	1 per Pge, if defined	Deleted when the collection is destroyed
	EcPlPREditor, EcPlOdMgr	PlTimeScheduled	Class that represents a Time Scheduled type PGE.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per Time Scheduled PGE	Deleted after PGE information is collected
	EcPlOdMgr	PlStandingOrderGranules	Collection of ASTER OnDemand input granules with DAR Ids associated with user requests.	EcPlOdMgr	not passed	EcPlOdMgr	1 per Standing Order with a matching DAR ID	Deleted after all relevant OnDemand orders are processed
	EcPlPREditor, EcPlOdMgr	PlSnapshotScheduled	Class that represents a Snapshot Scheduled type PGE.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per Snapshot Scheduled PGE	Deleted after PGE information is collected
	EcPlPREditor	PlRoutineArrival	Class that represents an input data ESDT that is ingested at regular, predictable time intervals.	EcPlPREditor	not passed	EcPlPREditor	1 per routinely arriving input granule	Deleted after Data Processing Request information is collected
	EcPlPREditor, EcPlOdMgr	PlOutputYield	Class that represents Data Processing Request outputs ESDTs.	EcPlPREditor, EcPlOdMgr	not passed not passed	EcPlPREditor, EcPlOdMgr	1 per DPR output ESDT	

Table 4-1. Memory Management Table (18 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
CSS	EcSbSubServer EcSbSubServer EcSbGui EcCsEmailParser EcCsMojoGateway EcCsMtMGateway EcCsRegistry	GlParameterList	A class that collects the general parameters of EMD.	EcSbSubServer EcSbSubServer EcSbGui EcCsEmailParser EcCsMojoGateway EcCsMtMGateway EcCsRegistry	EcSbCl not passed not passed not passed not passed not passed not passed	EcSbSubServer EcSbSubServer EcSbGui EcCsEmailParser EcCsMojoGateway EcCsMtMGateway EcCsRegistry	9 66 19 9 29 31 26	

Table 4-1. Memory Management Table (19 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcSbSubServer EcCsRegistry LoadingTool	RWDBMemTable	A Rogue Wave DB class that is a table of data residing in the program memory. After construction, an RWDBMemTable is no longer associated with a table in the database. An application can modify the data in an RWDBMemTable, but the changes are not propagated back to the database.	EcSbSubServer EcCsRegistry LoadingTool	not passed not passed not passed	EcSbSubServer EcCsRegistry LoadingTool	3 8 2	
	EcSeLoginProg EcSbSubServer LoadingTool EcCsRegistry	RWDBResult	A Rogue Wave DB class that represents a sequence of results whenever a database operation can potentially produce multiple SQL table expressions. Triggers that can cause results to be generated as a result of an INSERT, DELETE, or UPDATE statement.	EcSeLoginProg EcSbSubServer LoadingTool EcCsRegistry	not passed not passed not passed not passed	EcSeLoginProg EcSbSubServer LoadingTool EcCsRegistry	2 6 15 3	

Table 4-1. Memory Management Table (20 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcSeLoginProg EcSbSubServer EcCsRegistry LoadingTool EcCsIdNameServer	RWDBReader	A Rogue Wave DB class that provides row-by-row access to tabular data.	EcSeLoginProg EcSbSubServer EcCsRegistry LoadingTool EcCsIdNameServer	not passed not passed not passed not passed not passed	EcSeLoginProg EcSbSubServer EcCsRegistry LoadingTool EcCsIdNameServer	13 13 8 5 2	
	EcSbGui	EcClEventCollector	This class provides a collection mechanism for retrieving and manipulating multiple events.	EcSbGui	not passed	EcSbGui	1	
	EcSbGui	EcClSubscriptionCollector	This class provides a collection mechanism for retrieving and manipulating multiple subscriptions.	EcSbGui	not passed	EcSbGui	1	

Table 4-1. Memory Management Table (21 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcCsMtMGateway EcCsMojoGateway EcCsEmailParser EcSbSubServer	DsClESDTReferenceCollector	This class provides the primary interaction mechanism for client software. This class contains the specialized functions pertaining to management of state (the working collection on the server side) by mimicking that state on the client machine.	EcCsMtMGateway EcCsMojoGateway EcCsEmailParser EcSbSubServer	not passed not passed not passed not passed	EcCsMtMGateway EcCsMojoGateway EcCsEmailParser EcSbSubServer	1 1 1 1	
MSS	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	RWDBMemTable	A Rogue Wave DB class that is a table of data residing in the program memory. After construction, an RWDBMemTable is no longer associated with a table in the database. An application can modify the data in an RWDBMemTable, but the changes are not propagated back to the database.	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	not passed not passed not passed	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	2 1 6	

Table 4-1. Memory Management Table (22 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	RWDBResult	A Rogue Wave DB class that represents a sequence of results whenever a database operation may potentially produce multiple SQL table expressions. Triggers that can cause results to be generated as a result of an INSERT, DELETE, or UPDATE statement.	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	not passed not passed not passed	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	13 10 6	
	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	RWDBReader	A Rogue Wave DB class that provides row-by-row access to tabular data.	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	not passed not passed not passed	EcAcOrderSrvr EcMsAcRegUserSrvr MsCsSurveyMgrServer	16 13 29	

Table 4-1. Memory Management Table (23 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
INS	EcInGUI, EcInInitPasswd	CsSeCryptoDes::DesEncrypt	Used to encrypt passwords.	EcInGUI, EcInInitPasswd	not passed	EcInGUI	Three instances per update to the InExternalDataProviderInfo table (EcInGUI). One instance for EcInInitPasswd.	EcInInitPasswd does not delete the instance, but it is a test driver. The amount of memory allocated is the size of the encrypted password.
	EcInGUI, EcInPolling, EcInReqMgr	CsSeCryptoDes::DesDecrypt	Used to decrypt passwords.	EcInGUI, EcInPolling, EcInReqMgr	not passed not passed not passed	EcInGUI, EcInPolling, EcInReqMgr	Three instances per retrieval from the InExternalDataProviderInfo table.	The amount of memory allocated is the size of the encrypted password.
	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	InDataTypeTemplate	Used to access the InDataTypeTemplate table in the Ingest database. This table contains information about each data type that can be ingested.	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	not passed not passed not passed not passed	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	One instance	The memory is deallocated when the server comes down.

Table 4-1. Memory Management Table (24 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	InEDPAddressMap	Used to access the InEDPAddressMap table in the Ingest database. This table contains IP addresses, which can be mapped to external data provider names.	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	not passed not passed not passed not passed	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	One instance	The memory is deallocated when the server comes down.
	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	InCurrentDataTypeMap	Used to access the InCurrentDataTypeMap table in the Ingest database. This table contains the default version id for each data type that can be ingested.	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	not passed not passed not passed not passed	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	One instance	The memory is deallocated when the server comes down.
	EcInGUI	InValRequestState	Used to access the InValRequestState table in the Ingest database. This table contains the valid request states.	EcInGUI	not passed	EcInGUI	One instance	
	EcInGUI	InValDataGranuleState	Used to access the InValDataGranuleState table in the Ingest database. This table contains the valid granule states.	EcInGUI	not passed	EcInGUI	One instance	

Table 4-1. Memory Management Table (25 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	InConfig	Used to store configuration parameters for Ingest.	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	not passed not passed not passed not passed	EcInGUI, EcInGran, EcInPolling, EcInReqMgr	One instance	The memory is deallocated when the server comes down.
	EcInReqMgr	InGranuleServersQueue	Used to access the Ingest granule queue. EcInReqMgr uses the granule queue for dispatching granules to each EcInGran instance.	EcInReqMgr	not passed	EcInReqMgr	One instance	The memory is deallocated when the server comes down.
OWS	EcOwOgcEchoAdaptor	Z3950serverfacade.Session Z3950serverfacade.PresentService	Use to manage each connection between client and server, also manages present requests(some of which can be huge)	EcOwOgcEchoAdaptor	Not passed	EcOwOgcEchoAdaptor	One instance	The memory is deallocated when the server comes down, or when a Session is ended or when present request is terminated in any way.
	EcOwSynchronizer	DplInventory OwsGranule PerformanceTracker RegistrationException	Synchronizes the holdings in the WCS/WMS indexer with the holdings in the Data Pool	EcOwSynchronizer	Not passed	EcOwSynchronizer	One instance	The memory is deallocated when the server comes down.

Table 4-1. Memory Management Table (26 of 26)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcOwGeotiffConverter	GeotiffConversion GeotiffConversionException ObservableProxy	Converts hdf-eos files to geotiff format.	EcOwGeotiffConverter	Not passed	EcOwGeotiffConverter	One instance	The memory is deallocated when the server comes down.
OMS	EcOmOrderManager	OmSrClientDb OmSrDbInterface	Handles connection and queries to the database server.	EcOmOrderManager	Not passed	EcOmOrderManager	One instance	The memory is deallocated when the server comes down.
	EcOmOrderManager	OmSrDispatchQueue	Keeps track of requests for processing.	EcOmOrderManager	Not passed	EcOmOrderManager	Four instances	The memory is deallocated when the server comes down.
	EcOmOrderManager	OmServer	Main encapsulating class.	EcOmOrderManager	Not passed	EcOmOrderManager	One instance	The memory is deallocated when the server comes down.
	EcOmOrderManager	OmSrDistributionRequest	Stores information related to distribution requests.	EcOmOrderManager	Not passed	EcOmOrderManager	One instance per request	The memory is deallocated when the server comes down, or when present request is terminated in any way.
CLS	Not Applicable							
SSS	Not Applicable							
Toolkit	Not Applicable							

4.1 Data Server Subsystem Overview

The Data Server Subsystem (DSS) provides capabilities to store, search, retrieve, and distribute earth science and related data. The DSS provides data repositories and management capabilities to safely store data on a permanent basis. The DSS stages data needed for data processing by the Data Processing Subsystem (DPS) or for retrieval by users at external locations. The DSS organizes and stores its data by data types, and provides advanced search capabilities and processing services on those data types in support of earth science data users. The DSS distributes data to users either electronically or on physical media. It also includes administrative capabilities to operate and manage its hardware and software.

DSS functionality includes:

- The DSS stores (archives) the Ingest Subsystem (INS) ingested data and products created via the DPS
- The DSS receives service requests for data and data type services from external service requesters including the Data Management Subsystem (DMS), the Planning Subsystem (PLS), and the DPS
- The DSS provides (distributes) data in response to service requests, to the request originator, by means of electronic transfer. Alternatively, the subsystem can provide references to data as a Universal Reference (UR). The Product Distribution System (PDS) distributes data in response to originator service requests by means of physical media

Data Server Subsystem Context

Figure 4.1-1 is the context diagrams for the DSS. The diagrams show the events DSS sends to other SDPS or CSMS subsystems and the events sent to DSS.

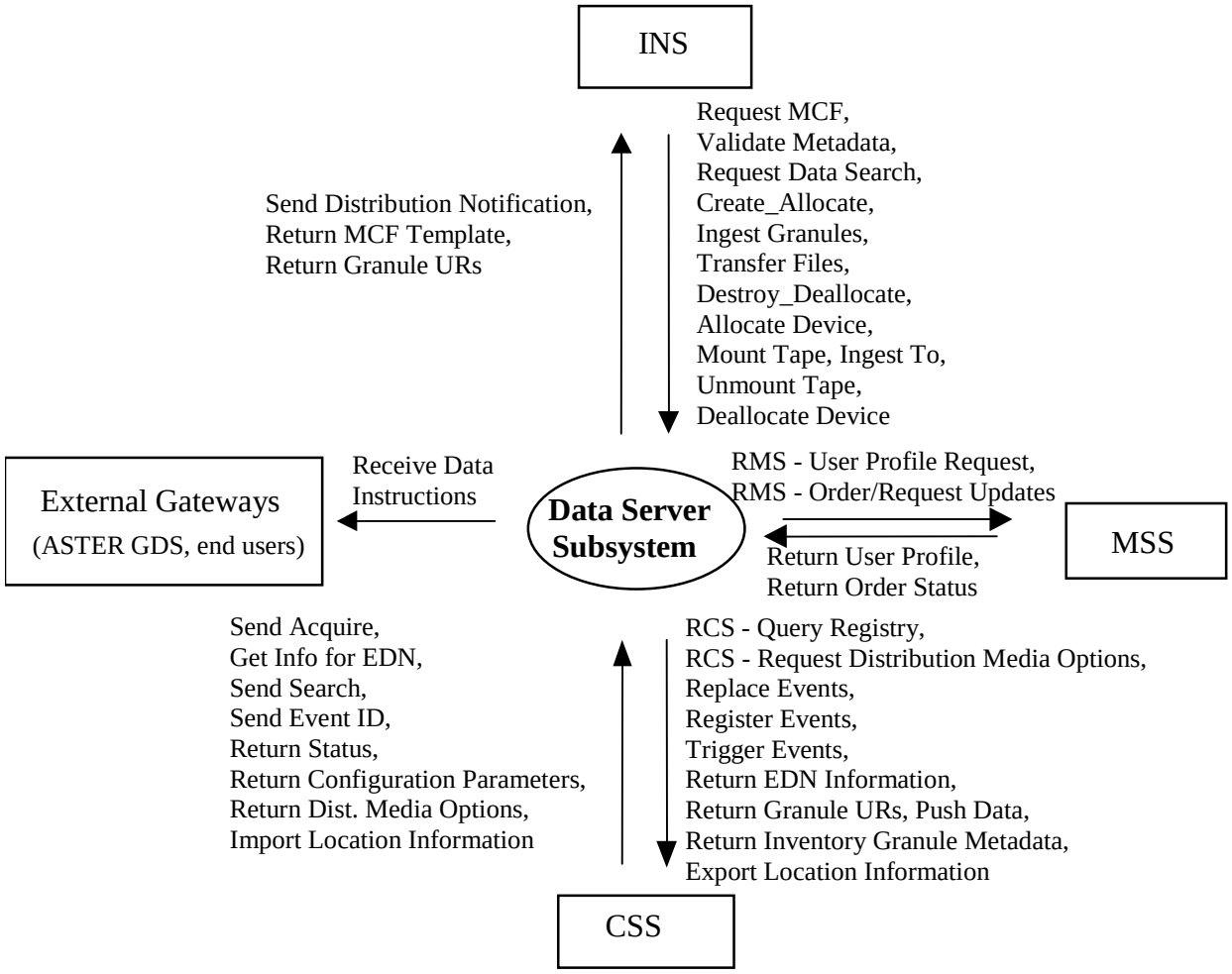


Figure 4.1-1. Data Server Subsystem Context Diagram

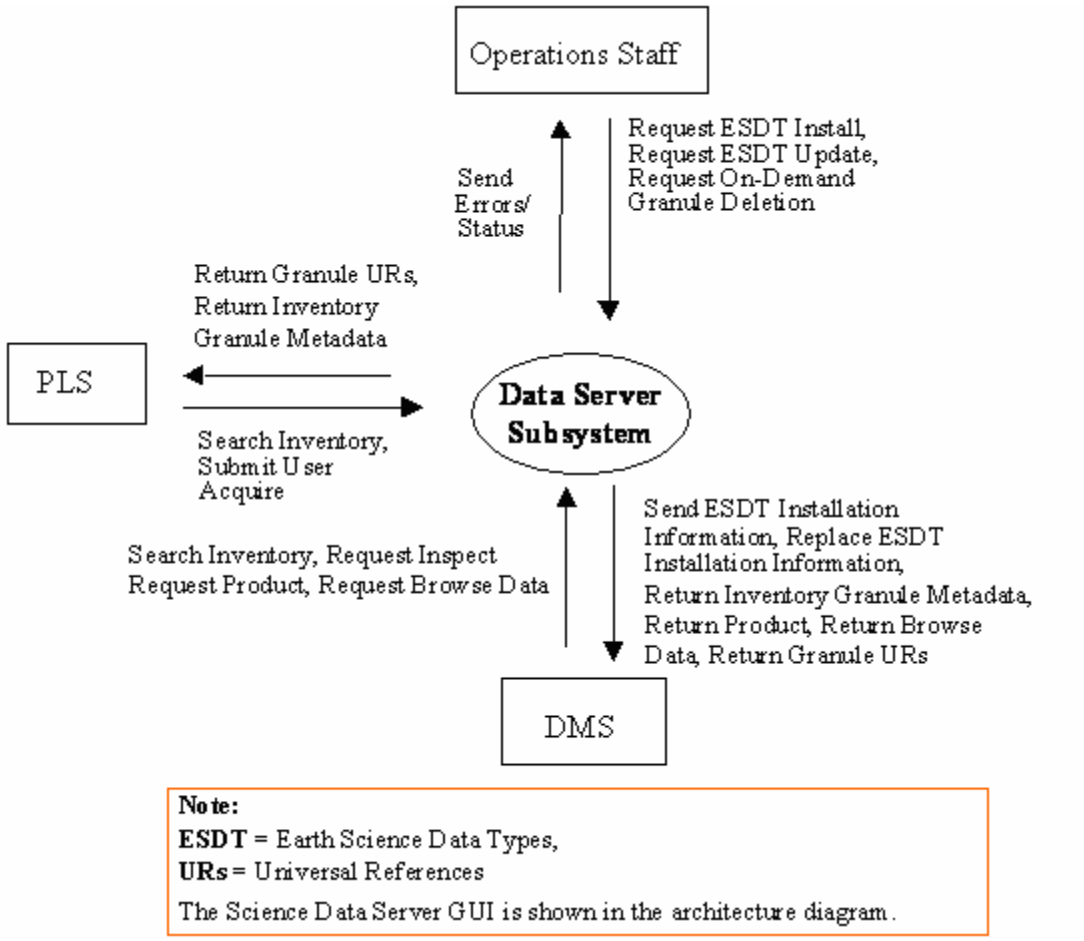


Figure 4.1-1. Data Server Subsystem Context Diagram (cont.)

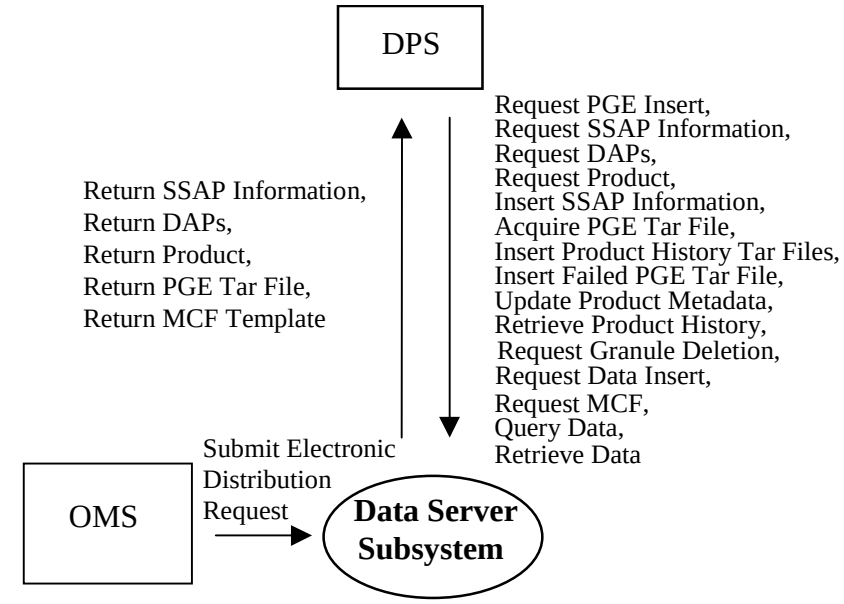


Figure 4.1-1. Data Server Subsystem Context Diagram (cont.)

Table 4.1-2 provides descriptions of the interface events shown in the Data Server Subsystem context diagrams.

Table 4.1-1. Data Server Subsystem Interface Events (1 of 5)

Event	Interface Event Description
Request MCF	The **INS** requests the Metadata Configuration File (MCF) from the DSS prior to a data insert request.
Validate Metadata	The **INS** populates the metadata files and sends a request to the DSS to validate the metadata files.
Request Data Search	The **INS** sends a search request to the DSS for a granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Create_Allocate	The **INS** sends requests to the DSS to allocate areas on the local staging disk to store ingested data.
Ingest Granules	The **INS** sends requests to the DSS to insert a particular file or files into the inventory and archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. For the INS, this data can be algorithms, Level 0 (L0) data, standard products, ancillary data, correlative data or calibration data. All data insert requests are sent to the DSS.
Transfer Files	The **INS** sends requests to the DSS to transfer (copy) data files to a staging disk.
Destroy_Deallocate	The **INS** sends requests to the DSS to deallocate a staging disk area (to remove an existing staging disk area from usage).
Allocate Device	The **INS** sends requests to the DSS to allocate peripheral devices for data ingesting.
Mount Tape	The **INS** sends requests to the DSS to load tapes to hardware peripherals for reading the tapes.
Ingest To	The **INS** sends requests to the DSS to copy files from peripheral resources to staging disk areas.
Unmount Tape	The **INS** sends requests to the DSS to unload and detach tapes from hardware peripherals after reading or writing to the tapes.
Deallocate Device	The **INS** sends requests to the DSS to deallocate the previously allocated media resource.
Request Management Services	The **MSS** provides a basic management library of services to the subsystems, implemented as client or server applications, using the CSS Process Framework. The basic management library of services includes: • **System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document. • **User Profile Request** - The MSS receives requests from the DSS for user profile information such as e-mail address and shipping address from authorized users to support their processing activities. • **Order/Request Updates** - The **DSS** (DDIST CSCI) interfaces with the Accountability Management Service Order/Request Tracking to create/update the user product order request such as media id, quantity and type.

Table 4.1-1. Data Server Subsystem Interface Events (2 of 5)

Event	Interface Event Description
Return User Profile	The **MSS** sends the user profile to the DSS for inventory searches, to request product orders, and to request subscriptions.
Return Order Status	The **MSS** provides order ids and order status information (Request ID for PLS) to the CLS, PLS and CSS for products requested by users.
Request Communications Support	The **CSS** provides a library of services available to each subsystem. The subsystem services required to perform specific assignments are requested from the CSS. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry • Request Distribution Media Options from the Configuration Registry
Replace Events	The **CSS** Subscription Server receives the updated subscription events with updated qualifiers for an Earth Science Data Type (ESDT) from the DSS when an ESDT is updated. This event replaces the original event in the CSS Subscription Server.
Register Events	The **CSS** Subscription Server receives the subscription events for an Earth Science Data Type (ESDT) from the DSS when an ESDT is installed into the system or when an ESDT is updated by adding additional events.
Trigger Events	The **CSS** is notified by the DSS (via an event trigger) when a subscription event occurs on an ESDT Service.
Return EDN Information	The **CSS** receives and uses the Expedited Data Set Notification (EDN) information from the DSS to send messages to users at the ASTER Ground Data System (GDS).
Return Granule URs	The **CSS** and **INS** receive Earth Science Data Type (ESDT) Universal References (URs) for the requested granules from the DSS.
Push Data	The DSS pushes data (i.e., EDS), via the FTP service and an FTP Daemon, to the **CSS (**ASTER DAR Gateway CSC) for data distribution per user request. A signal file is also sent to indicate the completion of the file transfer by particular ESDTs that function this way.
Return Inventory Granule Metadata	The **CSS** (MTMGW Server) receives the inventory granule metadata identifying the scene within the granule from the DSS based on an inventory search request.
Export Location Information	The DSS stores physical and logical server location information in the **CSS**.
Send Acquire	The **CSS** creates an “acquire” (instruction to obtain data) and sends it to the DSS via a CCS Middleware call. This is similar to the “Request Product” interface event, except it applies to EDOS expedited data.

Table 4.1-1. Data Server Subsystem Interface Events (3 of 5)

Event	Interface Event Description
Get Info for EDN	The **CSS** requests the EDN information from the DSS to send messages to users at the ASTER GDS.
Send Search	The **CSS** sends requests to the DSS, on behalf of the SIPS, to get qualified granule URs returned.
Send Event ID	The **CSS** sends Event Ids to the DSS when ESDTs are installed or when ESDTs are updated by adding additional events.
Return Status	The **CSS** returns status to the DSS to simply indicate that the request was received, not that the action succeeded.
Return Configuration Parameters	The DSS receives the configuration parameters and associated values from the Registry Server within the **CSS**.
Return Dist. Media Options	The DSS receives the requested distribution media options from the **CSS.**
Import Location Information	The DSS retrieves physical and logical server location information from the **CSS**.
Receive Data Instructions	The **External Gateways** or other external users receive instructions (assembled by the DSS) to send data via the CSS. The DSS requests data to be pushed, via the File Transfer Protocol (FTP) service and followed by a signal file, to the destination specified in an acquire instruction (by particular ESDTs that function this way).
Send Distribution Notification	The **INS** receives a distribution notification, via e-mail, from the DSS when data being distributed is to be ingested.
Return MCF Template	The **INS** receives the MCF template to populate, from the DSS, as part of the GetMCF service call.
Return Granule Urs	The **INS** receives the Earth Science Data Type (ESDT) Universal References (Urs) for the requested granules from the DSS.
Request ESDT Install	The **Operations Staff** sends ESDT installation information to the DSS for adding descriptor, Dynamic Link Library (DLL), and version id for a new Earth Science Data Type.
Request ESDT Update	The **Operations Staff** sends updated ESDT information to the DSS for adding updated descriptor and Dynamic Link Library (DLL) information for an existing ESDT.
Request On-Demand Granule Deletion	The **Operations Staff** sends requests to the DSS to delete science granules from the archive and inventory or just the archive. The associated PH, QA and Browse granules can also be deleted.
Get File Sizes	The **Product Distribution System (PDS)** Subsystem sends a request to the DSS to get file sizes.
Send ESDT Installation Information	The **DMS** Data Dictionary receives ESDT installation information from the DSS, whenever a new ESDT is installed. This data consists of Inventory and Collection level metadata.

Table 4.1-1. Data Server Subsystem Interface Events (4 of 5)

Event	Interface Event Description
Replace ESDT Installation Information	The **DMS** Data Dictionary receives updated ESDT information from the DSS, whenever an ESDT is updated. This data consists of updated Inventory and Collection level metadata.
Return Inventory Granule Metadata	The **DMS** and **PLS** receive the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the DSS.
Return Product	The **DMS** receives products from the DSS based upon a product request.
Return Browse Data	The **DMS** receives browse data associated with a particular granule from the DSS.
Return Granule Urs	The **DMS** and **PLS** receive Earth Science Data Type (ESDT) Universal References (Urs) for the granules from the DSS.
Search Inventory	The **DMS** and **PLS** send Inventory Search Requests to the DSS to search the SDPS Inventory (metadata).
Request Inspect	The **DMS** sends a request for an inspection of granule metadata to the DSS in support of a price estimate request.
Request Product	The **DMS** sends requests to the DSS for a product order from an external user to be distributed by the DSS.
Request Browse Data	The **DMS** submits requests for Browse data to the DSS to acquire reduced resolution products to support a product request.
Submit User Acquire	The **PLS** submits an acquire command to the DSS on behalf of the user. The user gets a response via the DSS upon data distribution.
Send Errors/Status	The **Operations Staff** receives error conditions and status of data distributions from the DSS.
Request PGE Insert	The **DPS** sends requests to the DSS to insert data that defines a Product Generation Executive (PGE) and allows it to be scheduled and executed.
Request SSAP Information	The **DPS** sends requests to the DSS for SSAP information, including names of existing SSAPs and the information associated with a specific SSAP.
Request DAPs	The **DPS** requests Data Archive Packages (DAPs) based on Urs from the DSS.
Request Product	The **DPS** sends requests, to the DSS, for particular data granules to be pushed, via the FTP service, onto the DPS science processor as input for data processing or for SSIT work.
Insert SSAP Information	The **DPS** sends requests to the DSS to insert SSAP information, via the DPS SSAP GUI by the Operations Staff, including SSAP name, SSAP version number, PGE name, PGE version number, and SSAP Acceptance Date.

Table 4.1-1. Data Server Subsystem Interface Events (5 of 5)

Event	Interface Event Description
Acquire PGE Tar File	The **DPS** acquires a tar file for any PGE not currently local to the science processor from the DSS. The executable is extracted from the tar file and used during PGE execution.
Insert Product History Tar Files	The **DPS** sends a request to the DSS to insert the PGE Production History Tar File resulting outputs for permanent archive after the PGE has successfully completed executing.
Insert Failed PGE Tar File	After an unsuccessful execution of a PGE, the **DPS** obtains the Tar file containing the PGE log files, core dump (if any), PCF and other files, and requests the files be inserted into the DSS for permanent archive.
Update Product metadata	The **DPS** (the Operations Staff) sends requests to update product metadata in the DSS using the QA Monitor GUI.
Retrieve Product History	The **DPS** (the Operations Staff using the QA Monitor GUI) submits requests to the DSS to transfer the Production History tar file from the Science Data archives to the user's host machine.
Request Granule Deletion	The **DPS** sends delete requests to the DSS for particular granules (interim data) in the metadata (the SDPS inventory).
Request Data Insert	The **DPS** sends insert requests to the DSS for a particular file or files (into the SDPS inventory and archives). Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. These files can be processing output, static files received with PGEs, PGE Tape Archive (TAR) files, Algorithm Packages (Aps), Science Software Archive Packages (SSAPs) or Delivered Algorithm Packages (DAPs), failed PGE tar files, or production history files.
Request MCF	The **DPS** requests the MCF from the DSS prior to a data insert request.
Query Data	The **DPS** submits search requests to the DSS for granules that match the user-supplied selection criteria: data type and begin/end date. Results are displayed to the user at a GUI in the DPS.
Retrieve Data	The **DPS** sends retrieval requests, to the DSS, for a particular data granuleId. The product is transferred (pushed), via the File Transfer Protocol (FTP) service, onto the DPS science processor and used as input for Product Generation Executive (PGE) processing or for Science Software Integration and Test (SSIT) work.
Submit Electronic Distribution Request	The **OMS** sends requests to the DSS to distribute EMD data electronically via an FTP Pull or FTP Push.
Return SSAP Information	The **DPS** receives lists of SSAPs and related information from the DSS.
Return DAPs	The DAPs are placed on a local **DPS** disk by the DSS.
Return Product	The data granules requested by the **DPS** are sent from the DSS.
Return PGE Tar File	After an unsuccessful execution of a PGE, the **DPS** obtains the Tar file containing the PGE log files, core dump (if any), Process Control File (PCF) and other files, and requests the files be inserted into the DSS for permanent archive.
Return MCF Template	The DSS provides the MCF template, to the **DPS**, to populate as part of the GetMCF service call.

Data Server Subsystem Structure

The DSS is three CSCIs:

- The Science Data Server (SDSRV) CSCI manages and provides user access to collections of non-document (non hard copy) earth science data, extracts and modifies data by request, accepts browse, search, and retrieval requests from users, and catalogs data insert requests from other SDPS or CSMS CSCIs, CSCs, and processes. The SDSRV CSCI manages earth science data as logical collections of related data, via interfaces independent of data formats and hardware configurations inherent in underlying storage technologies. The SDSRV manages interactive sessions with service requesters and informs the service requester of the availability of data and services via the IOS.

- The Data Distribution (DDIST) CSCI monitors and controls processing for distribution requests. Data Distribution processing consists of directing the STMGT CSCI to place data for distribution in working storage, creating packing lists, directing the STMGT CSCI to "FtpPush" or "FtpPull", and sending notifications for completed distribution requests. The DDIST CSCI accepts requests from the SDSRV CSCI. Electronic distribution can be requested via an FTP push or pull. With push, the DDIST CSCI uses network resources managed by the STMGT CSCI to transfer the data to a remote destination specified by the requester. For pull, the data is placed in an area managed by the STMGT CSCI, from which the requester can retrieve the data.

- The Storage Management (STMGT) CSCI stores and manages data, and retrieves data files from the archives for other science data processing software. The STMGT CSCI provides an interface to make implemented changes in new data storage technologies transparent to users and without interfering with EMD systems outside the STMGT CSCI. The STMGT CSCI performs quality assurance processing and files recovery services. The STMGT CSCI also provides management of storage resources and prepares data for distribution.

The Data Server Subsystem hardware consists of the following four Hardware Configuration Items (HWCIs):

- Access Control and Management

  The Access Control and Management HWCI (ACMHW) is hardware to support the Ingest and Data Server Subsystems' software to directly interact with users. The ACMHW provides a level of security by isolating other hardware items from external software access.

- Data Repository

  The Data Repository HWCI (DRPHW) is hardware to provide high-capacity storage for long-term storage of data files.

- Distribution and Ingest Peripherals

  The Distribution and Ingest Peripherals HWCI (DIPHW) is hardware to provide support to ingest and distribution via physical media.

Detailed information on hardware/software mapping, hardware diagrams, disk partitioning, etc., can be found in 920-TDx-00x, the 921-TDx-00x, and the 922-TDx-00x series of baseline documents. These documents are located at the web site http://pete.hitc.com/baseline/index.html and click on the Technical Documents button.

Use of COTS in the Data Server Subsystem

- RogueWave's Tools.h++

  The Tools.h++ class libraries provide libraries of object strings and collections. These class libraries are statically linked and delivered with the custom code installation.

- RogueWave's DBTools.h++

  The DBTools.h++ C++ class libraries interact with the Sybase database Structured Query Language (SQL) server and buffer the processes from the relational database used. These class libraries are statically linked and delivered with the custom code installation.

- Rogue Wave's Net.h++

  ToolsPro.h++ is a C++ class library, which includes the net.h++ class library, which provides an object-oriented interface to Inter-Process Communication (IPC) and network communication services. The Net.h++ framework enables developed code to be portable to multiple operating systems and network services. These libraries must be installed with the STMGT software to support interaction with other subsystems.

- Integrated Computer Solutions (ICS) Builder Xcessory

  The Builder Xcessory GUI builder tool modifies the displays. The Builder Xcessory generates the C++ code to produce the Maintenance Tool (Mtool) display at run time. There is no operational part of the Builder Xcessory needed at run-time.

- Sybase Adaptive Enterprise Server (ASE)

  The Sybase ASE provides the capabilities to insert, update and delete database contents. The Sybase ASE must be operational to execute search and insert requests for metadata.

- Boeing Autometric's Spatial Query Server

  The Spatial Query Server (SQS) provides the capability to store and search spatial metadata. SQS has spatial indexing to search on spatial metadata for the SDSRV.

- Sybase Open Client / CT_LIB

  The Sybase Open Client provides access between DSS custom code and the Sybase ASE DBMS.

- University of Illinois' Hierarchical Data Format (HDF)

  HDF provides EOS extended capabilities for sub-setting services with the SDSRV CSCI.

- University of Colorado's Object Description Language (ODL)

  ODL provides a general architecture, independent means of passing metadata files between subsystems.

- CCS Middleware Client

  CCS Middleware Client provides DSS with communications between other subsystems. CCS Middleware can reside on one or both sides of the interface. An instance must be installed on the platform where DSS resides. Although the CCS Middleware Client is part of CSS, this COTS product must be installed for DSS to run in the SDPS operational and test environment.

Error Handling and processing

EcUtStatus is a class used throughout the EMD custom code for general error reporting. It is almost always used as a return value for functions and allows detailed error codes to be passed back up function stacks.

DsShError is a Science Data Server specific class used mainly for exception handling.

DsShErrorDetails is a Science Data Server class that can be used to convert error details (in an EcUtStatus object) into more meaningful text messages.

The Science Data Server uses two main mechanisms for error handling.

1. Return Values

   Functions can return an EcUtStatus object, which can be used to indicate a general success/failure status. Also, more detailed information on the exact reason for the failure can be provided. For example, a granule cannot be acquired because it has restricted access privileges. This is the most widely used mechanism within the Science Data Server and in general these errors get propagated back up to the top-level functions with ALOG error messages being generated along the way.

2. Exceptions

   Some functions (for example, class constructors) cannot return values to indicate success or failure. These functions may throw exceptions, usually instances of the DsShError class. These errors are usually caught by other functions at a low level and converted into EcUtStatus return values (as described in 1).

In addition, the DsShErrorDetails class can be used to map error values (as contained in an EcUtStatus object) into text messages. This enables better reporting of errors in the Science Data Server logs.

Currently, the Science Data Server client interface only supports returning error messages back to client programs, along with a generic success/failed status.

For writing messages to the Applications Log (ALOG), the following functions are used:

DsLgLogError sends a message to the ALOG at severity level 1. For example, DsLgLogError (“DsMdMetadataCheckpoint1”, “Bad granule UR”);

DsLgLogWarning sends a message to the ALOG at severity level 2. For example, DsLgLogWarning (“DsMdMetadataCheckpoint2”, “Unable to retrieve granule metadata”);

DsLgLogInformational sends a message to the ALOG at severity level 3. For example, DsLgLogInformational (“DsMdMetadataCheckpoint3”, “Failed to construct granule”);

For writing messages to the debug log, the following macros are used:

PF_STATUS writes a message at a “log level” of 1 to the debug log. For example, PF_STATUS {cerr << “Issue rpc to STMGT” << endl;}

PF_VERBOSE writes a message at a “log level” of 2 to the debug log. For example, PF_VERBOSE {cerr << “Request received from client” << endl;}

PF_DEBUG writes a message at a “log level” of 3 to the debug log. For example, PF_DEBUG {cerr << “Saved request to database” << endl;}

The class EcUtStatus is used to hold the actual error number. The EcUtStatus object is returned to the DDIST clients when the request is complete.

The DDICT CSCI uses two main mechanisms for error handling.

1. Return Values

   Functions can return an EcUtStatus object, which can be used to indicate a general success/failure status. Also, more detailed information on the exact reason for the failure can be provided. This is the most widely used mechanism within the DDICT and in general these errors get propagated back up to the top-level functions with ALOG error messages being generated along the way.

2. Exceptions

   Some functions (for example, class constructors) cannot return values to indicate success or failure. These functions may throw exceptions, usually instances of the DsShError class. These errors are usually caught by other functions at a low level and converted into EcUtStatus return values (as described in 1).

In addition, the DsShErrorDetails class can be used to map error values (as contained in an EcUtStatus object) into text messages. This enables better reporting of errors in the DDIST logs.

Currently, the DDIST client interface only supports returning error messages back to client programs, along with a generic success/failed status.

For writing messages to the Applications Log (ALOG), the following functions are used:

EcLgLogError sends a message to the ALOG at severity level 1. For example,

```
catch (DsShError& err)
{ EcLgLogError (“DsDdSchedulerExecuteFunctionError” , 0, err.GetMsg());}
```

EcLgLogInformational sends a message to the ALOG at severity level 3. For example,

```
EcLgLogInformational (“ConfigVarMissing”,status.GetLogMessageLink(),
  “EcCUtRpcClientIDConfigTag var not set in Config File.”);
```

For writing messages to the debug log, the following macros are used:

PF_STATUS writes a message at a “log level” of 1 to the debug log. For example,

```
PF_STATUS {cerr << “DsDdCBCache::Create Creating first instance.” << endl;}
```

PF_VERBOSE writes a message at a “log level” of 2 to the debug log. For example,

```
PF_VERBOSE {cerr << “Calling DsStArchive::Create. ReqID | FullArchiveID:\n”
        << myRequestID <<” | “ <<  FullArchiveID << “ and\n”
        << “RpcID = “ << RpcId_CR.AsString() << endl;}
```

PF_DEBUG writes a message at a “log level” of 3 to the debug log. For example,

```
PF_DEBUG {cerr << “DsDdDistRequestS::SetPriority.  Request: “
        <<  myRequestID << “ Priority: “ << Priority <<  endl;}
```

The Storage Management (STMGT) CSCI uses the following classes to deal with errors:

The class EcUtStatus is used to hold the actual error number when an error occurs. The EcUtStatus object is returned to the STMGT clients when the request is complete. The class DsStErrorDetails is used to extract the error type and severity information from the EcUtStatus class.

The class DsStLogging is used throughout the STMGT code to control the writing of error messages to the application log files and debug log files. This class uses the EMD standard logging functions and macros. Errors can also be logged to the DsStEventLog table within the STMGT database. These errors can be viewed with the STMGT GUI. These errors are periodically removed by the DsStRequestManager Server / GRCleanup Stored Procedure.

Here is an example of how logging is used:

```
EcUtStatus status;
.
.
.
if (somethingWentWrong)
{
   ourDsStLogging.LogAndSetError(status, DsEStInternalSybError,


“DBIF:FetchQueryResult: ct_fetch failed”);
}
```

The program EcDsStErrorFilesGenerator is used to load the various error codes used by STMGT into the STMGT database.

4.1.1 Science Data Server Software Description

4.1.1.1 Science Data Server Functional Overview

The SDSRV CSCI provides the SDPS with a catalog of Earth Science Data holdings, and the Earth Science Data Type services that operate on the data. The SDSRV CSCI provides a catalog of metadata describing the archived data holdings of the SDPS and provides mechanisms to acquire the data from the archive. The SDSRV CSCI also provides data type services on the catalog and a data reduction or sub-setting and reformatting services.

4.1.1.2 Science Data Server Context

Figure 4.1-2 the SDSRV CSCI context diagrams. The diagrams show the events sent to the SDSRV CSCI and the events the SDSRV CSCI sends to other CSCIs. The events have been grouped by CSCI including the Storage Management and Data Distribution functions of the DSS.

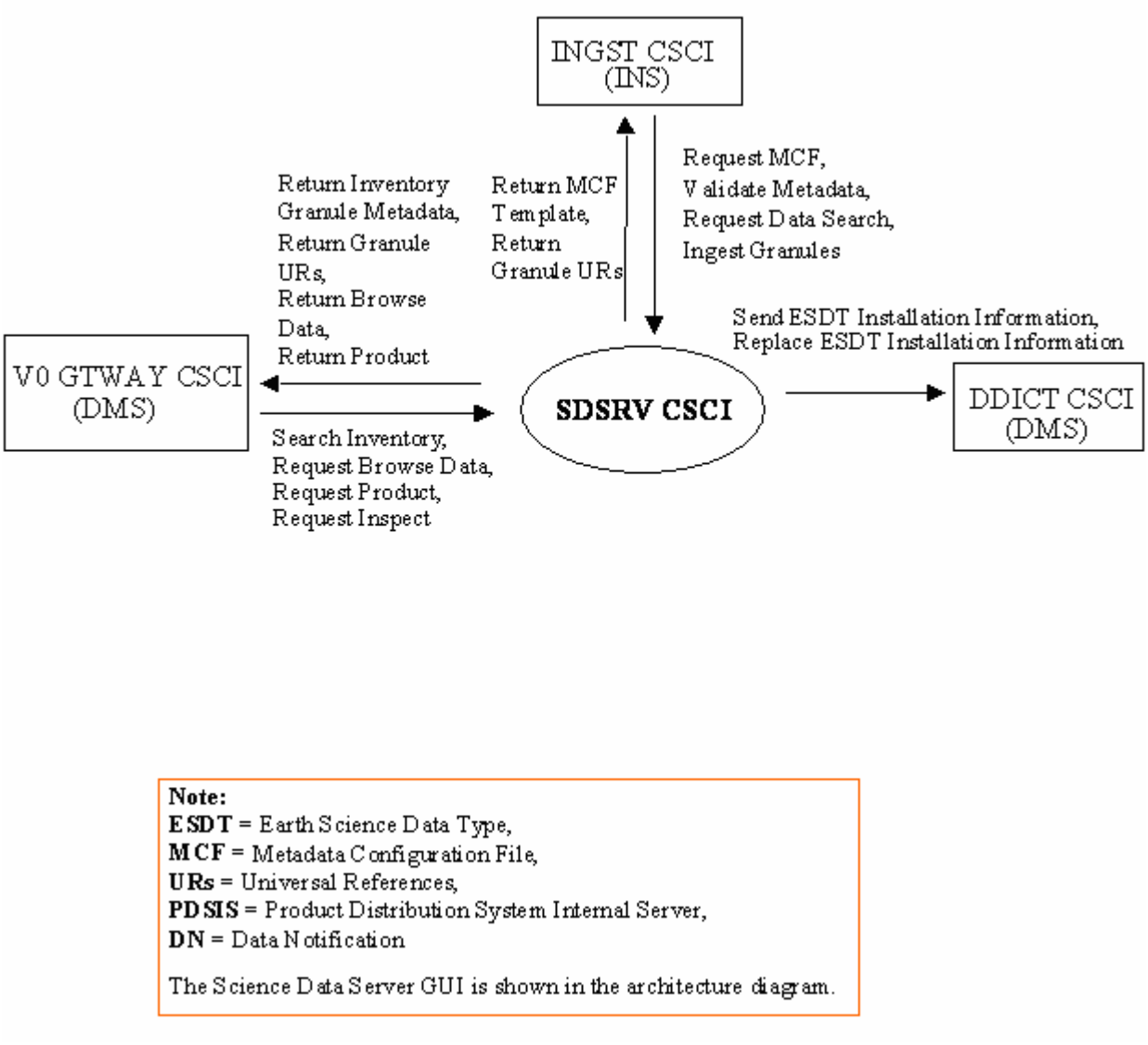
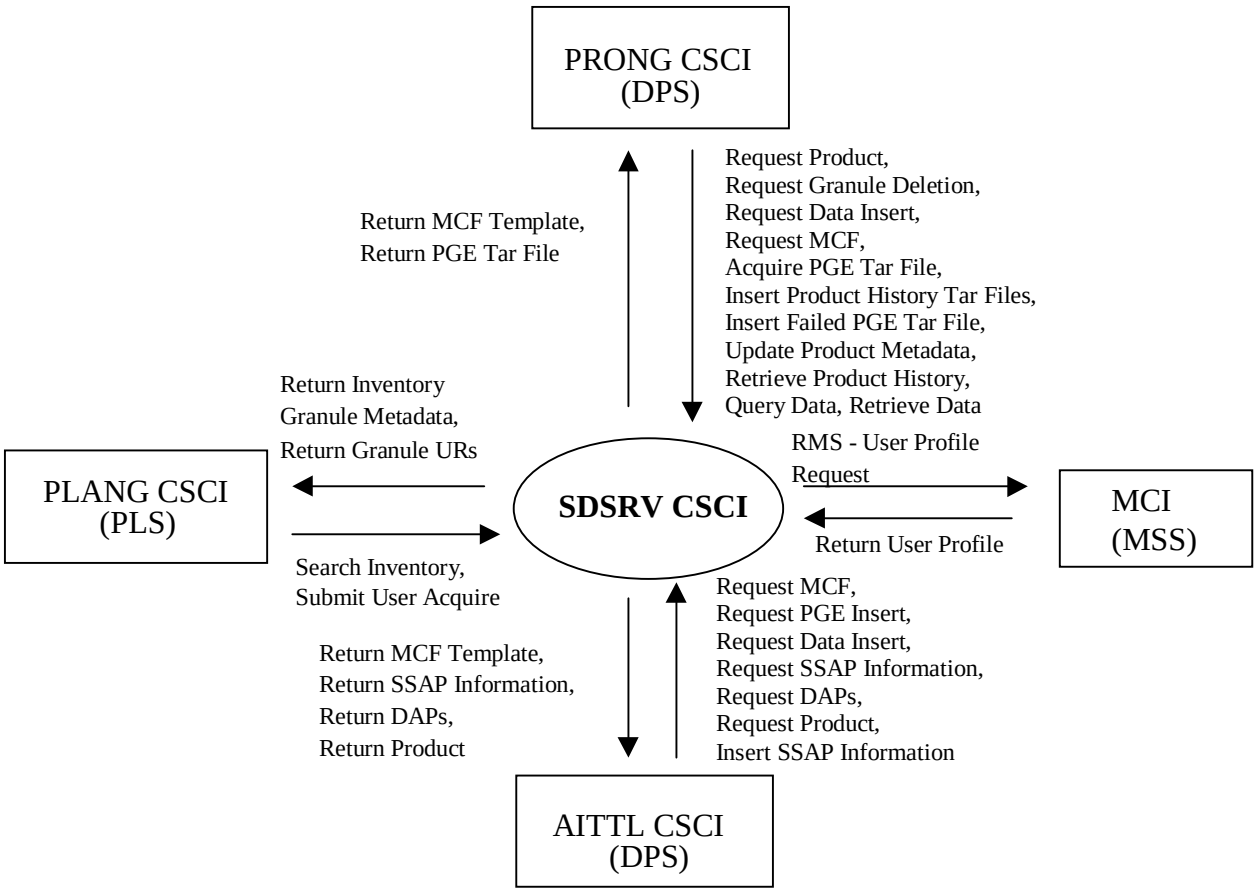


Figure 4.1-2. SDSRV CSCI Context Diagram



Note:
RMS = Request Management Services,
URs = Universal References,
PGE = Product Generation Executable,
SSAP = Science Software Archive Package,
DAPs = Delivered Algorithm Packages,
MCF = Metadata Configuration File

The Science Data Server GUI is shown in the architecture diagram.

Figure 4.1-2. SDSRV CSCI Context Diagram (cont.)

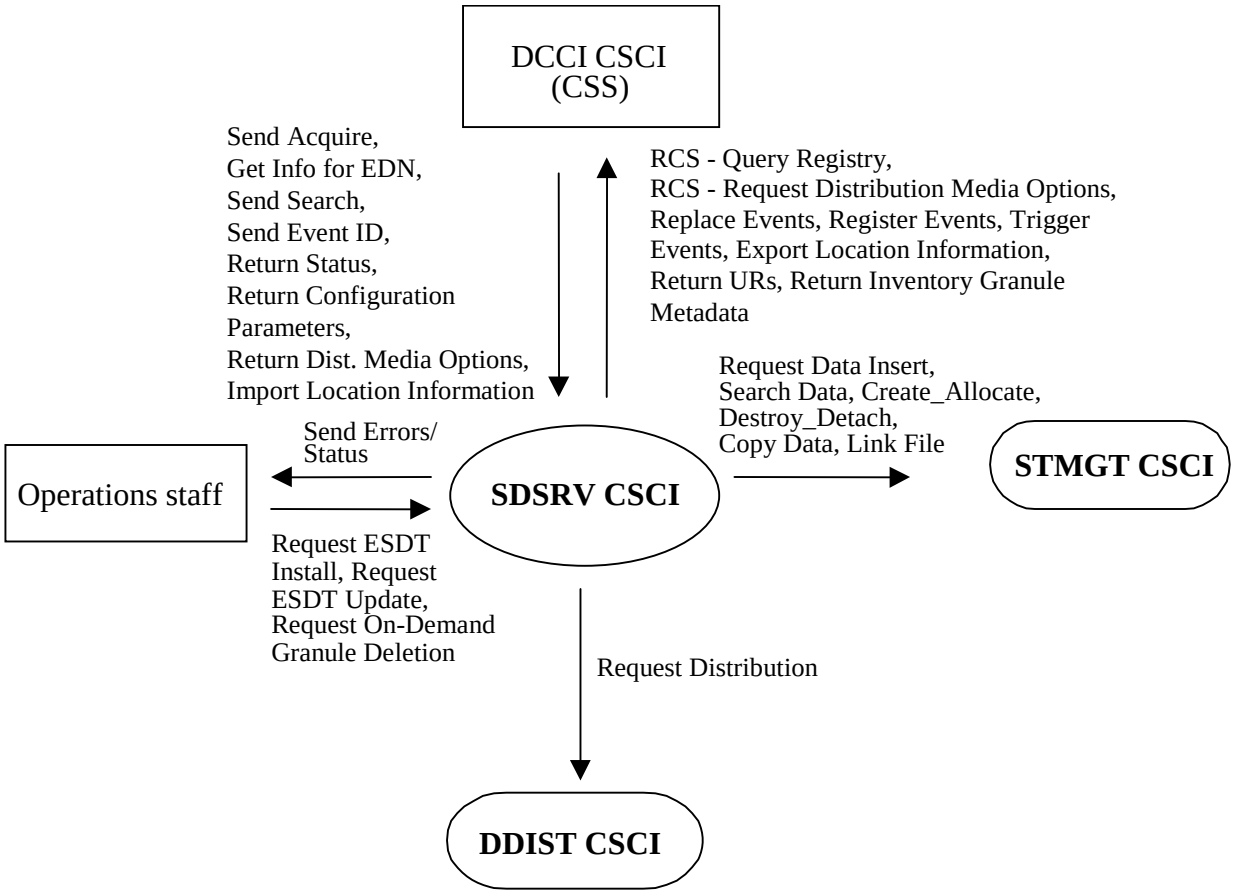


Figure 4.1-2. SDSRV CSCI Context Diagram (cont.)

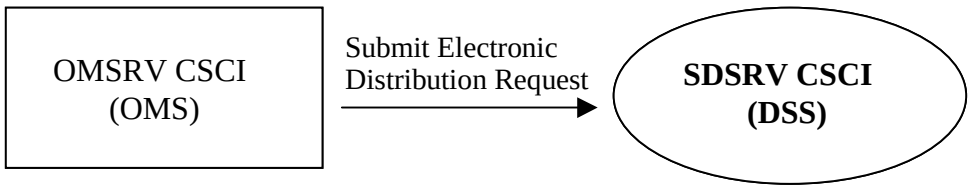


Figure 4.1-2. SDSRV CSCI Context Diagram (cont.)

Table 4.1-3 provides descriptions of the interface events shown in the SDSRV CSCI context diagrams.

Table 4.1-2. SDSRV CSCI Interface Events (1 of 6)

Event	Interface Event Description
Request MCF	The **INGST CSCI** requests the Metadata Configuration File (MCF) template, from the SDSRV CSCI, for each input or output data type, respectively, prior to a data insert request. The SDSRV CSCI provides the MCF information as part of the GetMCF service call.
Validate Metadata	The **INGST CSCI** populates the metadata files and sends a request to the SDSRV CSCI to validate the metadata files.
Request Data Search	The **INGST CSCI** sends a search request to the SDSRV CSCI for a granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Ingest Granules	The **INGST CSCI** sends requests to the SDSRV CSCI to insert a particular file or files into the inventory and archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. For the INGST CSCI, this data can be algorithms, Level 0 (L0) data, standard products, ancillary data, correlative data or calibration data. All data insert requests are sent to the STMGT CSCI from the SDSRV CSCI.
Send ESDT Installation Information	The **DDICT CSCI** receives ESDT installation information from the SDSRV CSCI, whenever a new ESDT is installed. This data consists of Inventory and Collection level metadata.
Replace ESDT Installation Information	The **DDICT CSCI** receives updated ESDT information from the SDSRV CSCI, whenever an ESDT is updated. This data consists of Inventory and Collection level metadata. The updated information replaces the ESDT information in the DDICT CSCI.
Get File Sizes	The **PDSIS CSCI** sends a request to the SDSRV CSCI to get file sizes.
Submit Acquire	The **PDSIS CSCI** submits an acquire request to the SDSRV CSCI to retrieve data granules of interest.
Search Inventory	The **V0 GTWAY CSCI** sends inventory search requests to the SDSRV CSCI to search the SDPS Inventory (metadata).
Request Browse Data	The **V0 GTWAY CSCI** submits requests for browse data to the SDSRV CSCI to acquire reduced resolution products to support a product request.
Request Product	The **V0 GTWAY CSCI** sends requests to the SDSRV CSCI for a product order from an external user to be distributed by the DDIST CSCI upon receipt of the data from the STMGT CSCI.
Request Inspect	The **V0 GTWAY CSCI** sends a request for an inspection of granule metadata to the SDSRV CSCI in support of a price estimate request.
Return Inventory Granule Metadata	The **V0 GTWAY CSCI** receives the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the SDSRV CSCI.

Table 4.1-2. SDSRV CSCI Interface Events (2 of 6)

Event	Interface Event Description
Return Granule URs	The **V0 GTWAY CSCI** and **INGST CSCI** receive Earth Science Data Type (ESDT) Universal References (URs) for the granules from the SDSRV CSCI.
Return Browse Data	The **V0 GTWAY CSCI** receives browse data associated with a particular granule from the SDSRV CSCI.
Return Product	The **V0 GTWAY CSCI** receives products from the SDSRV CSCI based upon a product request.
Return MCF Template	The **INGST CSCI** receives the template to populate as part of the GetMCF service call to the SDSRV CSCI.
Request Product	The **PRONG CSCI** sends requests, to the SDSRV CSCI, for particular data granules to be pushed, via the FTP service, onto the DPS science processor as input for data processing or for SSIT work.
Request Granule Deletion	The **PRONG CSCI** sends delete requests to the SDSRV CSCI for particular granules (interim data) in the metadata (the SDPS inventory).
Request Data Insert	The **AITTL** and **PRONG CSCIs** send insert requests to the SDSRV CSCI for a particular file or files (into the SDPS inventory and archives). Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. These files can be processing output, static files received with PGEs, PGE Tape Archive (TAR) files, Algorithm Packages (APs), Science Software Archive Packages (SSAPs) or Delivered Algorithm Packages (DAPs), failed PGE tar files, or production history files.
Request MCF	The **PRONG** and **AITTL CSCIs** request the Metadata Configuration File (MCF), from the SDSRV CSCI, for each input or output data type, respectively, prior to a data insert request. The SDSRV CSCI provides the MCF information as part of the GetMCF service call. Also, the **PRONG CSCI** can request from the SDSRV CSCI the MCF for a particular ESDT short name prior to a data insert request.
Acquire PGE Tar File	The **PRONG CSCI** acquires a tar file for any PGE not currently local to the science processor from the SDSRV CSCI. The executable is extracted from the tar file and used during PGE execution.
Insert Product History Tar Files	The **PRONG CSCI** sends a request to the SDSRV CSCI to insert the PGE Production History Tar File resulting outputs for permanent archive after the PGE has successfully completed executing.
Insert Failed PGE Tar File	After an unsuccessful execution of a PGE, the **PRONG CSCI** obtains the Tar file containing the PGE log files, core dump (if any), PCF and other files, and requests the files be inserted into the SDSRV CSCI for permanent archive.
Update Product Metadata	The **PRONG CSCI** (the Operations Staff using the QA Monitor GUI) sends requests to update product metadata in the SDSRV CSCI.

Table 4.1-2. SDSRV CSCI Interface Events (3 of 6)

Event	Interface Event Description
Retrieve Product History	The **PRONG CSCI** (the Operations Staff using the QA Monitor GUI) submits requests to the SDSRV CSCI to transfer the Production History tar file from the Science Data archives to the user's host machine.
Query Data	The **PRONG CSCI** submits requests of this type to the SDSRV CSCI. It searches the archive for granules that match the user-supplied selection criteria: data type and begin/end date. Results are displayed to the user at a GUI in the PRONG CSCI.
Retrieve Data	The **PRONG CSCI** sends retrieval requests, to the SDSRV CSCI, for a particular data granuleId. The product is transferred (pushed), via the FTP service, onto the DPS science processor and used as input for PGE processing or for SSIT work.
Request Management Services	The **MCI** provides a basic management library of services to the CSCIs, implemented as client or server applications, using the DCCI CSCI Process Framework. The basic management library of services includes: • **System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document • **Request User Profile** - The MCI receives requests from the SDSRV CSCI for user profile information such as e-mail address and shipping address from authorized users to support their processing activities
Return User Profile	The **MCI** sends the user profile to the SDSRV CSCI for inventory searches, to request product orders, and to request subscriptions.
Request PGE Insert	The **AITTL CSCI** sends requests to the SDSRV CSCI to insert data that defines a PGE and allows it to be scheduled and executed.
Request SSAP Information	The **AITTL CSCI** sends requests to the SDSRV CSCI for SSAP information, including names of existing SSAPs and the information associated with a specific SSAP.
Request DAPs	The **AITTL CSCI** requests DAPs based on URs from the SDSRV CSCI.
Request Product	The **AITTL CSCI** sends requests to the SDSRV CSCI to push data granules, via the FTP service, onto the DPS science processor as input for data processing or SSIT work.
Insert SSAP Information	The **AITTL CSCI** (the Operations Staff via the SSAP GUI) sends requests to the SDSRV CSCI to insert SSAP information, including SSAP name, SSAP version number, PGE name, PGE version number, and SSAP Acceptance Date.
Return MCF Template	The **AITTL** and **PRONG CSCIs** receive the MCF template to populate, from the SDSRV CSCI, as part of the GetMCF service call.
Return SSAP Information	The SDSRV CSCI sends lists of SSAPs and related information to the **AITTL CSCI**.
Return DAPs	The SDSRV CSCI places the DAPs on a local AITTL CSCI disk.
Return Product	The data granules requested by the **PRONG** and **AITTL CSCIs** are sent from the SDSRV CSCI.

Table 4.1-2. SDSRV CSCI Interface Events (4 of 6)

Event	Interface Event Description
Search Inventory	The **PLANG CSCI** sends requests to the SDSRV CSCI to search the SDPS Inventory (archives).
Submit User Acquire	The **PLANG CSCI** submits an acquire command to the SDSRV CSCI on behalf of the user. The user gets a response via the DDIST CSCI upon data distribution.
Return Inventory Granule Metadata	The **PLANG CSCI** receives the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the DSS.
Return Granule URs	The **PLANG CSCI** receives Earth Science Data Type (ESDT) Universal References (URs) for the granules from the SDSRV CSCI.
Return PGE Tar File	After an unsuccessful execution of a PGE, the **PRONG CSCI** obtains the Tar file containing the PGE log files, core dump (if any), PCF and other files, and requests the files be inserted into the SDSRV CSCI for permanent archive.
Request Communications Support (RCS)	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI/CSC. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry • Request Distribution Media Options from the Configuration Registry
Replace Events	The **DCCI CSCI** (Subscription Server) receives updated subscription events with modified qualifiers for an Earth Science Data Type (ESDT) from the SDSRV CSCI, when an ESDT is updated. This event replaces the original event in the DCCI CSCI.
Register Events	The **DCCI CSCI** (Subscription Server) receives the subscription events for an Earth Science Data Type from the SDSRV CSCI, when an ESDT is installed into the system or when an ESDT is updated by adding additional events.
Trigger Events	The **DCCI CSCI** receives notification from the SDSRV CSCI (via an event trigger) when a subscription event occurs on an Earth Science Data Type Service.
Export Location Information	The SDSRV CSCI stores physical and logical server location information in the **DCCI CSCI**.
Return URs	The **DCCI CSCI** receives Earth Science Data Type (ESDT) Universal References (URs) for the requested granules from the SDSRV CSCI.
Return Inventory Granule Metadata	The **DCCI CSCI** (MTMGW Server CSC) receives the inventory granule metadata identifying the scene within the granule from the SDSRV CSCI based on an inventory search request.

Table 4.1-2. SDSRV CSCI Interface Events (5 of 6)

Event	Interface Event Description
Request Data Insert	The **STMGT CSCI** receives data insert requests from the SDSRV CSCI for a particular file or files (into the SDPS inventory and archives). Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. These files can be processing output, static files received with PGEs, PGE Tape Archive (TAR) files, Algorithm Packages (APs), Science Software Archive Packages (SSAPs) or Delivered Algorithm Packages (DAPs), failed PGE tar files, or production history files.
Search Data	The **STMGT CSCI** receives search requests from the SDSRV CSCI for granules corresponding to a particular ESDT short name and version, which has a particular local granule id.
Create_Allocate	The **STMGT CSCI** receives requests from the SDSRV CSCI to allocate areas on the local staging disk to store "ingested" data or output files from routine data processing or SSIT work.
Destroy_Detach	The **STMGT CSCI** receives requests from the SDSRV CSCI to detach from a staging disk area (lose access to an existing staging disk area owned by another process).
Copy Data	The **STMGT CSCI** receives requests from the SDSRV and DDIST CSCIs to copy data within staging disks and between staging disks.
Link File	The **STMGT CSCI** receives SDSRV and DDIST CSCIs send requests to the STMGT CSCI to link files from read-only cache to a staging disk specified in the request.
Request Distribution	The **DDIST CSCI** receives distribution requests from the SDSRV CSCI for various categories of data. The distribution services on those data are essentially identical for all data categories.
Request ESDT Install	The **Operations Staff** sends ESDT installation information to the SDSRV CSCI for adding the descriptor, Dynamic Link Library (DLL), and version id for a new Earth Science Data Type.
Request ESDT Update	The **Operations Staff** sends updated ESDT information to the SDSRV CSCI for adding updated descriptor and Dynamic Link Library (DLL) information for an existing ESDT.
Request On-Demand Granule Deletion	The **Operations Staff** sends requests to the SDSRV CSCI to delete science granules from the archive and inventory or just the archive. The associated PH, QA and Browse granules can also be deleted.
Send Errors/Status	The **Operations Staff** receives error conditions and status of data distribution from the SDSRV CSCI (for acquires that are synchronous) or the DDIST CSCI (if the acquire is asynchronous and the DDIST CSCI gets the request).
Send Acquire	The **DCCI CSCI** creates an “acquire” (instruction to obtain data) and sends it to the SDSRV CSCI. This is similar to the “Request Product” interface event, except it applies to EDOS expedited data.
Get Info for EDN	The **DCCI CSCI** receives the Expedited Data Set Notification (EDN) information from the SDSRV CSCI, by request, and uses the EDN information to send messages to users at the ASTER GDS.

Table 4.1-2. SDSRV CSCI Interface Events (6 of 6)

Event	Interface Event Description
Send Search	The **DCCI CSCI** sends requests to the SDSRV CSCI, on behalf of the SIPS, to get qualified granule URs returned.
Send Event ID	The **DCCI CSCI** sends Event Ids to the SDSRV CSCI when ESDTs are installed or when ESDTs are updated by adding additional events.
Return Status	The **DCCI CSCI** returns status to the SDSRV CSCI to simply indicate that the request was received, not that the action succeeded.
Return Configuration Parameters	The SDSRV CSCI receives the configuration parameters and associated values from the Registry Server from the **DCCI CSCI**.
Return Dist. Media Options	The SDSRV CSCI receives the requested distribution media options from the **DCCI CSCI.**
Import Location Information	The SDSRV CSCI retrieves physical and logical server location information from the **DCCI CSCI**.
Send Electronic Distribution Request	The **OMSRV CSCI** sends requests to the SDSRV CSCI to distribute EMD data electronically via an FTP Pull or FTP Push.

4.1.1.3 Science Data Server Architecture

Figure 4.1-3 is the SDSRV CSCI architecture diagrams. The diagrams show the events sent to the SDSRV CSCI processes and the events the SDSRV CSCI processes send to other processes.

The Science Data Server (SDSRV) CSCI is six processes: four SDPS custom developed processes and two COTS processes. The four SDPS custom developed processes are the Science Data Server (EcDsScienceDataServer), the Hierarchical Data Format (HDF) EOS Server (EcDsHdfEosServer) [Note: multiple HDF Server processes can be defined.], and the Science Data Server GUI (EcDsSdSrvGui), and the Granule Deletion Administration Tool (EcDsBulkDelete.pl). The COTS processes are the Sybase ASE and the Spatial Query Server (SQS). The SDSRV CSCI uses the Sybase ASE Database Management System (DBMS) for SDPS Inventory and Configuration data storage. The server holds Earth Science Data Type configuration information and the data catalog for all the archived products found at a DAAC.

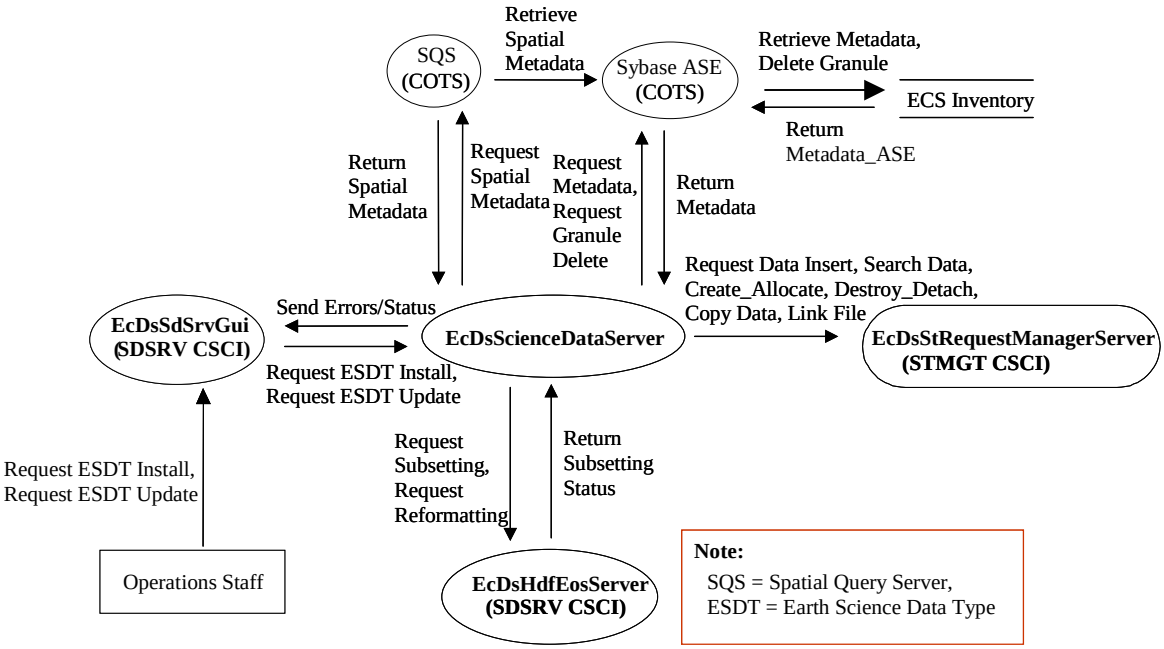


Figure 4.1-3. SDSRV CSCI Architecture Diagram

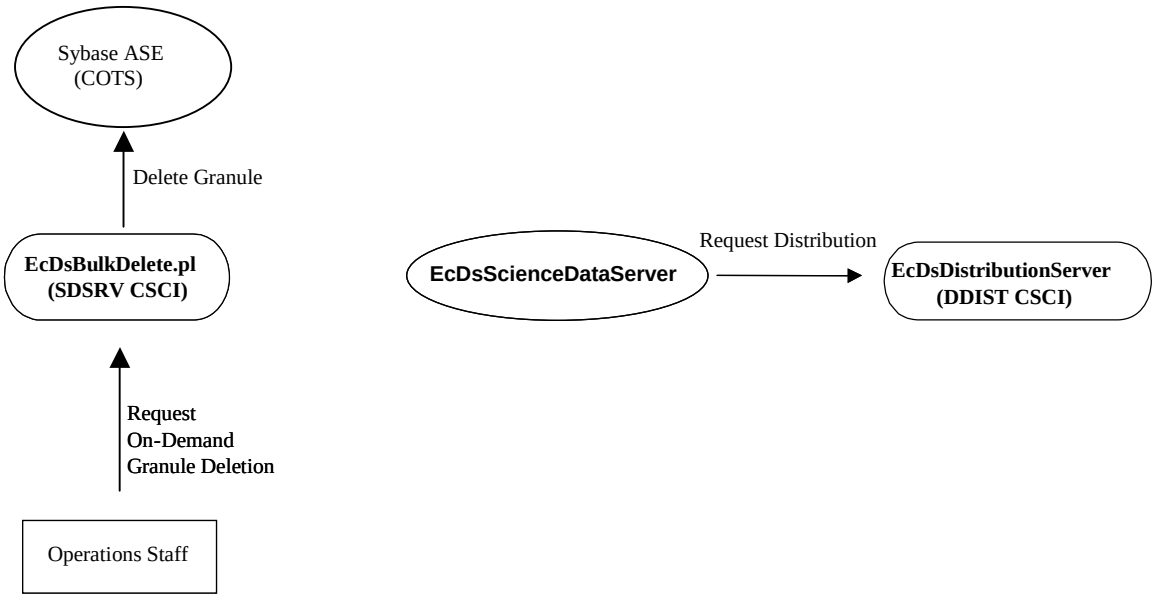


Figure 4.1-3. SDSRV CSCI Architecture Diagram (cont.)

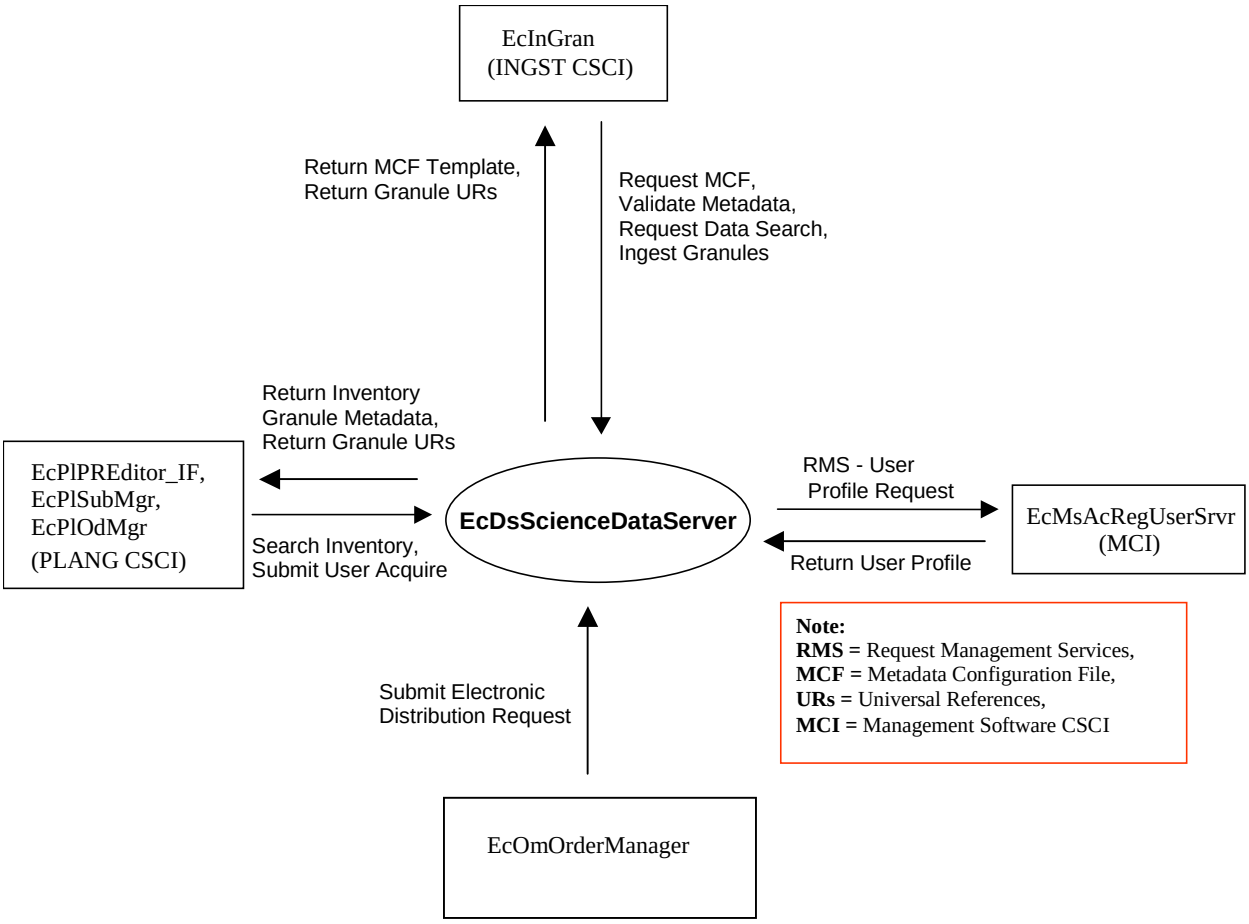


Figure 4.1-3. SDSRV CSCI Architecture Diagram (cont.)

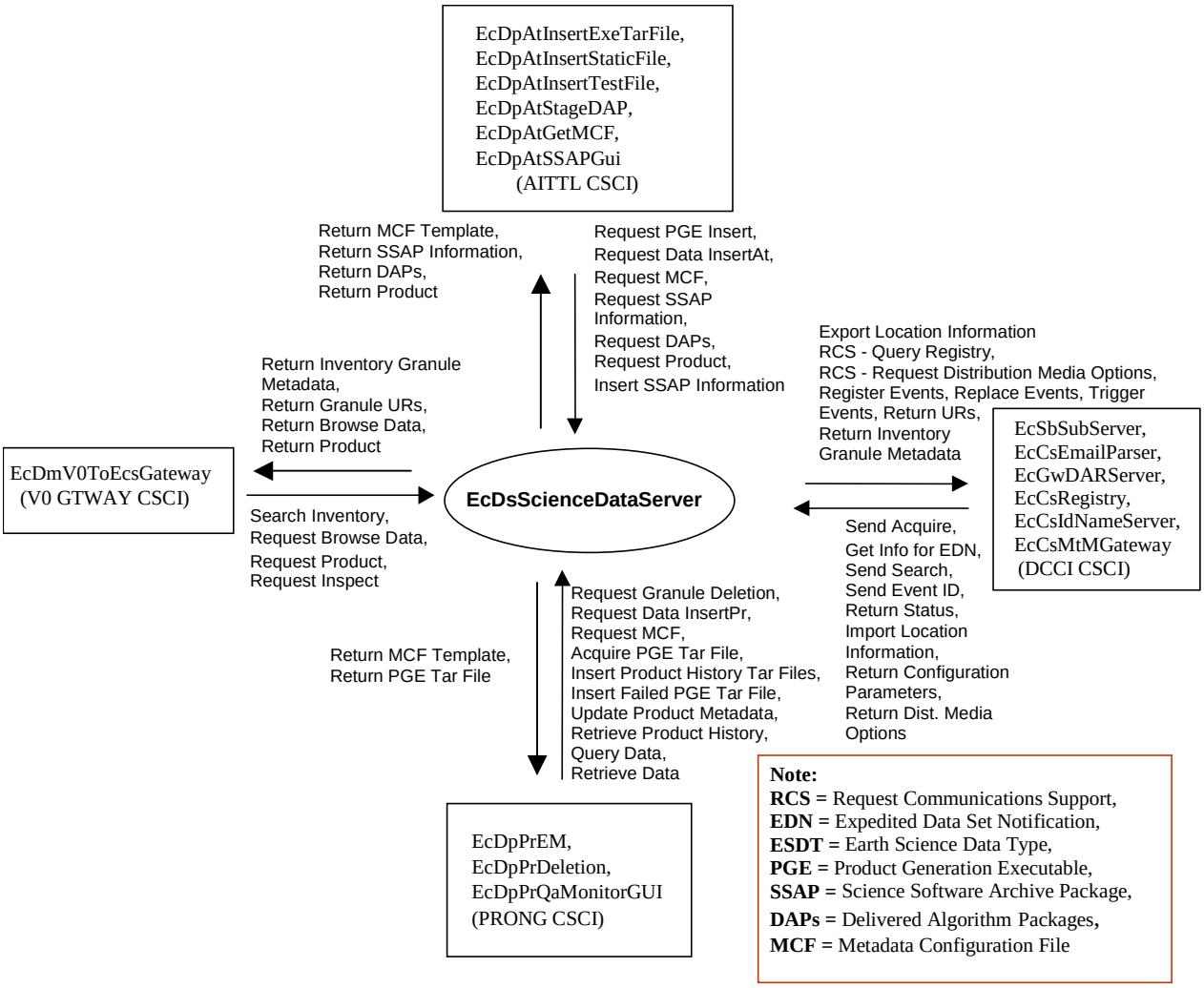


Figure 4.1-3. SDSRV CSCI Architecture Diagram (cont.)

4.1.1.4 Science Data Server Process Descriptions

Table 4.1-4 provides descriptions of the processes shown in the SDSRV CSCI architecture diagrams.

Table 4.1-3. SDSRV CSCI Processes (1 of 2)

Process	Type	Hardware CI	COTS / Developed	Functionality
EcDsScienceDataServer	Server	ACMHW	Developed	The EcDsScienceDataServer server manages collections of earth science and related data, and service requests for the storage, search, retrieval, and manipulation of data within those collections. The science data server performs the following functions: • Manages earth science data as logical collections of related data, using interfaces independent of any data formats and hardware configurations provided by underlying storage technologies, • Manages interactive sessions with users, • Manages the processing of service requests from the DMS (V0 Gateway), providing a variety of services on earth science and related data, • Issues requests to the STMGT and DDIST CSCIs to perform storage and distribution services in support of processing service requests, • Manages the processing of service requests from the INS and DPS to "insert" data for long-term storage and access, • Manages the processing of service requests from the DPS to provide data to be used as input for data processing, • Provides subscription events and event triggers to the CSS subscription server • Issues commands to the CSS EcSbSubServer to replace subscription events • Provides sub-setting requests to the EcDsHdfEosServer for scene sub-setting • Provides sub-setting requests to the EcDsHdfEosServer for compound sub-setting • Provides reformatting request to the EcDsHdfEosServer for conversion from HDFEOS to HDF The EcDsScienceDataServer supports: • Single requests, one at a time • Multiple concurrent requests • Asynchronous request processing • Request processing buffered from SOCKET Communication Call threads • Multiple threads within a single user session

Table 4.1-3. SDSRV CSCI Processes (2 of 2)

Process	Type	Hardware CI	COTS / Developed	Functionality
EcDsHdfEosServer	Server	DRPHW	Developed	The EcDsHdfEosServer provides science data sub-setting capabilities for earth science data configured with a sub-setting service. EcDsHdfEosServer supports: • Single requests, one at a time • Asynchronous request processing • Request processing buffered from SOCKET Communication Call threads • Sub-setting requests of spatial (floating scene or fixed scene)/band/temporal sub-setting • Reformatting requests from HDFEOS to HDF
EcDsSdSrvGui	GUI	ACMHW	Developed	The EcDsSdSrvGui provides an operator interface for: • Receiving descriptor files and dynamic link libraries (dll) for configuring ESDTs into the EcDsScienceDataServer • Monitoring active EcDsScienceDataServer requests • Updating ESDT information in the EcDsScienceDataServer The EcDsSdSrvGui supports: • Single requests, one at a time
EcDsBulkDelete.pl	Command Line Utility	ACMHW	Developed	The EcDsBulkDelete.pl provides a command line operator interface for: • Deleting granules in the EcDsScienceDataServer and the EcDsStArchiveServer • Deleting granules in the EcDsScienceDataServer
Sybase ASE	Server	ACMHW	COTS	Provides the management of spatial data types of an earth science catalog of metadata for the SDPS. Includes capabilities for searching and storing the catalog.
Spatial Query Server (SQS)	Server	ACMHW	COTS	Provides the capability to manage spatial data types of earth science catalog metadata for the SDPS (including specialized spatial searches).

4.1.1.5 Science Data Server Process Interface Descriptions

Table 4.1-5 provides descriptions of the interface events shown in the SDSRV CSCI architecture diagram for Figure 4.1-3.

Table 4.1-4. SDSRV CSCI Process Interface Events (1 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Metadata	One per request to store, search, delete, update, Earth Science Metadata	*Process:* Sybase ASE (COTS)	*Processes:* EcDsScienceDataServer *Library:* DsDb *Class:* DsDbInterface via the Sybase ASE, SQS, and Sybase Open Client COTS	The **Sybase ASE** receives requests from the EcDsScienceDataServer to store, search, delete, or update Earth Science Metadata. The results are sent back to the EcDsScienceDataServer. The M&O Staff must manually change a configured parameter in order for the EcDsScienceDataServer to communicate directly with the Sybase ASE.
Request Granule Delete	One granule per request	*Process:* Sybase ASE (COTS)	*Process:* EcDsScienceDataServer *Library:* DsDb *Class:* DsDbInterface	The **Sybase ASE** receives requests to delete granules from the EcDsScienceDataServer.
Retrieve Metadata	One to many per metadata request	Data Tables within the EMD Inventory	*Process:* Sybase ASE *Libraries (Sybase):* Libtcl.so Libtli.so Libsybdb.so *Class:* Sybase Open Client /ct_lib	The **Sybase ASE** retrieves metadata from the EMD Inventory database and returns the metadata to the SQS or the EcDsScienceDataServer.
Delete Granule	One per request	Data Tables within the EMD Inventory	*Process:* EcDsScienceDataServer *Library:* DsDb *Class:* DsDbInterface	The **Sybase ASE** requests granules to be deleted from the EMD Inventory data tables.
Return Metadata_ASE	One to many per metadata request	*Process:* Sybase ASE (COTS)	Data Tables within the EMD Inventory	The **EMD Inventory data tables** provide the metadata or spatial metadata to the Sybase ASE.

Table 4.1-4. SDSRV CSCI Process Interface Events (2 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Metadata	Per request	*Process:* Sybase ASE (COTS)	*Process:* EcDsScienceDataServer *Libraries:* DpPrDssIF, DsDb *Class:* DsDbInterface	Metadata results are sent back to the EcDsScienceDataServer via the SQS. The results include the status of the Sybase ASE commands.
Request Data Insert	One per data insert request from EcDpPrEM or EcInGran	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsGeESDT	The **EcDsStRequestManagerServer** receives data insert requests from the EcDsScienceDataServer for data to be stored in the SDPS inventory.
Search Data	One granule per search request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The **EcDsStRequestManagerServer** receives search requests from the EcDsScienceDataServer for a granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Create_Allocate	One allocation per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT	The **EcDsStRequestManagerServer** receives requests from the EcDsScienceDataServer to allocate areas on the local staging disk to store data for distribution.
Destroy_Detach	One detach per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT	The **EcDsStRequestManagerServer** receives requests from the EcDsScienceDataServer to detach from (lose access) to an existing staging disk area owned by another process.

Table 4.1-4. SDSRV CSCI Process Interface Events (3 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Copy Data	One file copy per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT	The **EcDsStRequestManagerServer** receives requests from the EcDsScienceDataServer to copy data within staging disks and between staging disks.
Link File	One link per file in a request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT	The **EcDsStRequestManagerServer** receives requests from the EcDsScienceDataServer to link files from the read-only cache to a staging disk and from one staging disk to another.
Return Subsetting status	One per sub-setting request	*Process:* EcDsScienceDataServer *Library:* DsCsSh *Classes:* DsCsConformant DsCsNonConformant	*Process:* EcDsHdfEosServer *Classes:* DsESDTLsL70R DsESDTLsL70RWRS	The **EcDsHdfEosServer** sends the status of sub-setting requests to the EcDsScienceDataServer.
Request Subsetting	One per request to reduce resolution	*Process:* EcDsHdfEosServer *Library:* DsCsSh *Classes:* DsCsConformant DsCsNonConformant	*Process:* EcDsScienceDataServer *Classes:* DsESDTLsL70R DsESDTLsL70RWRS	The **EcDsHdfEosServer** receives requests from the EcDsScienceDataServer to reduce the resolution of an archived earth science data product using configured services for spatial (floating scene or fixed scene) / band/temporal reduction.
Request Reformatting	One per request to reformat the data	*Process:* EcDsHdfEosServer *Libraries:* DsDc, DsCsSh *Class:* DsCsNonConformantImp	*Process:* EcDsScienceDataServer *Classes:* DsESDTLsL70R DsESDTLsL70RWRS	The **EcDsHdfEosServer** receives requests from the EcDsScienceDataServer to convert HDFEOS format to HDF format.

Table 4.1-4. SDSRV CSCI Process Interface Events (4 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Request ESDT Install	One per new ESDT installation	*Process:* EcDsScienceDataServer *Library:* DsAd *Class:* DsAdDataTypeCollector	Operations Staff *Process:* EcDsSdSrvGui *Class:* DsGuSdDatatype	The **Operations Staff** sends ESDT installation information for adding the descriptor, dynamic link library (dll), and version ID for a new ESDT to the EcDsScienceDataServer, via the EcDsSdSrvGui.
Request ESDT Update	One per new ESDT update	*Process:* EcDsScienceDataServer *Library:* DsAd *Class:* DsAdDataTypeCollector	Operations Staff *Process:* EcDsSdSrvGui *Class:* DsGuSdDatatype	The **Operations Staff** sends updated ESDT information, via the EcDsSdSrvGui, to the EcDsScienceDataServer, for adding updated descriptor and dynamic link library (dll) information for an existing ESDT.
Send Errors/Status	Once per distribution request completion	Operations Staff *Process:* EcDsSdSrvGui *Class:* DsGuErrorDialog	*Process:* EcDsScienceDataServer *Library:* EcUt *Class:* EcLgErrorMsg	The **Operations Staff** receives, via the **EcDsSdSrvGui**, error conditions and status of ESDT installs and updates from the EcDsScienceDataServer (for acquires that are synchronous).
Return Spatial Metadata	Per request	*Process:* EcDsScienceDataServer *Libraries:* DpPrDssIF, DsDb *Class:* DsDbInterface	*Process:* SQS (COTS)	Metadata results are sent back to the EcDsScienceDataServer via the SQS. The results include the status of the Sybase ASE and SQS Server commands. The SQS handles the translation of spatial metadata data types (understood by the EcDsScienceDataServer and SQS) to relational data types (understood by SQS and the Sybase ASE) and vice versa.

Table 4.1-4. SDSRV CSCI Process Interface Events (5 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Retrieve Spatial Metadata	One per request	*Process:* Spatial Query Server (SQS) [COTS]	*Processes:* EcDsScienceDataServer *Library:* DsDb *Class:* DsDbInterface via the Sybase ASE, SQS, and Sybase Open Client COTS	The **SQS** sends requests to the Sybase ASE to store, search, delete, or update Earth Science Metadata. The results are sent back to the SQS. The default configuration is for the EcDsScienceDataServer to communicate via the SQS to the EMD inventory.
Request Spatial Metadata	One per request to store, search, delete, or update spatial Earth Science Metadata.	*Process:* SQS (COTS)	*Process:* EcDsScienceDataServer *Libraries:* DpPrDssIF, DsDb *Class:* DsDbInterface	The **SQS** receives requests from the EcDsScienceDataServer to store, search, delete, or update spatial Earth Science Metadata in the EMD inventory database.
Request Distribution	One per distribution request	*Process:* EcDsDistributionServer *Library:* DsDdC *Class:* DsDdRequestMgrC	*Process:* EcDsScienceDataServer *Library:* DsSr *Class:* DsSrWorkingCollection	The **EcDsDistributionServer** receives distribution requests from the EcDsScienceDataServer for various categories of data. The distribution services are essentially identical for all data categories.
Request On-Demand Granule Deletion	Per Operations Staff request	*Process:* Sybase ASE (COTS)	Operations Staff *Process:* EcDsBulkDelete.pl	The **Operations Staff** sends requests, via the **Granule Delete Tool**, to the Sybase ASE database server to delete science granules from the archive and inventory or just the archive. The associated PH, QA and Browse granules can also be deleted. A physical delete or a Delete From Archive can be performed.

Table 4.1-4. SDSRV CSCI Process Interface Events (6 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Delete Granule	One per request	*Script:* EcDsBulkDelete.pl	*Process:* Sybase ASE (COTS)	The Sybase ASE database server deletes the requested granule from the archive and inventory or just the archive.
Request MCF	One per set of external data received by EMD	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	The **EcInGran** process requests the Metadata Configuration File (MCF) from the EcDsScienceDataServer, prior to a data insert request. The EcDsScienceDataServer provides the MCF information as part of the GetMCF service call.
Validate metadata	One per data insert request.	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	The **EcInGran** process populates the metadata files and sends requests to the EcDsScienceDataServer to validate the metadata files.
Request Data Search	One per input pointer in metadata or per granule pointer in linkage file	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	The **EcInGran** process sends a search request to the EcDsScienceDataServer for a granule corresponding to particular ESDT short name and version, which has a particular local granule id.

Table 4.1-4. SDSRV CSCI Process Interface Events (7 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Ingest Granules	One per data insert request from EcInGran	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsGeESDT	Process: EcInGran *Library:* InPreprocess *Class:* InDataServerInsertionTask	The **EcInGran** process sends requests to the EcDsScienceDataServer to insert a particular file or files into the archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. This data can be algorithms, Level 0 (L0) data, standard products, ancillary data, correlative data or calibration data.
Request Management Services (RMS)	One service per request	N/A	N/A	The **EcMsAcRegUserSrvr** provides a basic management library of services to the processes, implemented as client or server applications, using the DCCI CSCI process Framework. The basic management library of services includes: (See the table cells below.)
RMS (cont.)	At system startup or shutdown and for restarts	*Process:* EcMsAcRegUserSrvr	DAAC unique startup scripts	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document.
RMS (cont.)	One profile per request	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Class:* MsAcUsrProfile	*Process:* EcDsScienceDataServer *Library:* DsSr *Class:* DsSrManagedServer	**User Profile Request -** The EcMsAcRegUserSrvr receives requests from the EcDsScienceDataServer for user profile information such as e-mail address and shipping address from authorized users to support their processing activities.

Table 4.1-4. SDSRV CSCI Process Interface Events (8 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Return User Profile	One per request	*Process:* EcDsScienceDataServer *Library:* DsSr *Class:* DsSrManagedServer	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Classes:* MsAcUsrProfile, RWPortal	The **EcMsAcRegUserSrvr** returns the user profile to the EcDsScienceDataServer.
Submit Electronic Distribution Request	One order per user	*Process:* EcDsScienceDataServer	*Process:* EcOmOrderManager *Executable:* EcOmSrCLI *Library:* DsClientSideLibs *Class:* OmSdsrvIf	The **EcOmOrderManager** sends requests to the EcDsScienceDataServer to distribute EMD data electronically via an FTP Pull or FTP Push.

Table 4.1-4. SDSRV CSCI Process Interface Events (9 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Search Inventory	One per query	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	*Processes:* EcPlPREditor_IF, EcPlSubMgr, EcPlOdMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The **EcPlPREditor_IF**, **EcPlSubMgr** and **EcPlOdMgr** processes create two types of queries. One type only has the ESDT short name and data start and stop times and the other type also includes spatial coordinates. The EcPlPREditor_IF process queries when the predicted data is available. The EcPlPREditor_IF process creates an ESDT Reference from an UR after receiving an ESDT Reference from a query. The EcPlSubMgr process creates an ESDT Reference from an UR after receiving a subscription notification or receiving an ESDT reference from a query. The EcPlSubMgr process queries when predicted data is not available. The EcPlOdMgr process creates an ESDT Reference from an UR after receiving a subscription notification or receiving an ESDT reference from a query. The EcPlOdMgr process queries when predicted data is not available.

Table 4.1-4. SDSRV CSCI Process Interface Events (10 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Submit User Acquire	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsCl Request, DsClCommand, DsClESDTReferenceCollector	*Process:* EcPlOdMgr *Library:* PlCore2 *Classes:* PlPrActivator, DpPrDSSInterface	The **EcPlOdMgr** submits an acquire command to the EcDsScienceDataServer on behalf of the user. The user gets a response via the EcDsDistributionServer upon data distribution.
Return Inventory Granule Metadata	One set of metadata per granule	*Processes:* EcPlPREditor_IF, EcPlSubMgr, EcPlOdMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The EcDsScienceDataServer returns metadata information about the granule(s) being inspected to the **EcPlPREditor_IF**, **EcPlSubMgr** and **EcPlOdMgr** processes.
Return Granule URs	One per input pointer in metadata or per granule pointer in linkage file	*Processes:* EcPlPREditor_IF, EcPlSubMgr, EcPlOdMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface *Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The EcDsScienceDataServer returns ESDT References for granules to satisfy the query to the **EcPlPREditor_IF**, **EcPlSubMgr** and **EcPlOdMgr** processes. The EcDsScienceDataServer returns the granule URs for the granules requested in the data search to the **EcInGran** process.
Return MCF Template	One per set of external data received by EMD	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	The EcDsScienceDataServer provides the MCF information as part of the GetMCF service call to the **EcInGran** process.

Table 4.1-4. SDSRV CSCI Process Interface Events (11 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Request PGE Insert	One per insert request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Processes:* EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile *Library:* PlCore2 *Classes:* DpAtDsrv, PlResourceRequirement	The **EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile,** and **EcDpAtInsertTestFile** send PGE insert requests to the EcDsScienceDataServer for data that defines a PGE and allows it to be scheduled and executed.
Request Data InsertAt	One per data insert request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsGeESDT	*Processes:* EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile *Library:* DpAtDsrv *Class:* DpAtDsrv	The **EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile**, and **EcDpAtInsertTestFile**, processes send requests to the EcDsScienceDataServer to insert a particular file or files into the archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. For the EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, and EcDpAtInsertTestFile processes these files can be processing output, static files received with PGEs, PGE Tape Archive (TAR) files, Algorithm Packages (APs), Science Software Archive Packages (SSAPs), or Delivered Algorithm Packages (DAPs), failed PGE tar files, or production history files.
Request MCF	One per set of external data received by the EMD	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	*Processes:* EcDpPrEM, EcDpAtGetMCF *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The **EcDpAtGetMCF** and **EcDpPrEM** processes request the Metadata Configuration File (MCF) from the EcDsScienceDataServer, prior to a data insert request.

Table 4.1-4. SDSRV CSCI Process Interface Events (12 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Request SSAP Information	One per SSAP information request	*Process:* EcDsScienceDataServer *Library:* DsCI *Classes:* DsCIESDTReferenceCollector, DsCIRequest, DsCICommand	*Process:* EcDpAtSSAPGui *Libraries:* DpAtSSAP, DpAtDsrv *Classes:* DpAtSSAPManager, DpAtDsrv	The **EcDpAtSSAPGui** sends requests to the EcDsScienceDataServer for information about SSAPs, including names of existing SSAPs and the components associated with a specific SSAP.
Request DAPs	One per DAPs request	*Process:* EcDsScienceDataServer *Library:* DsCI *Classes:* DsCIESDTReferenceCollector, DsCIRequest, DsCICommand	*Process:* EcDpAtStageDAP *Library:* DpAtDsrv *Class:* DpAtDsrv	The **EcDpAtStageDAP** requests DAPs from the SDSRV Archives based on the UR. In response, the DAPs are returned and stored on the local AITTL disk.
Request Product	One per product order request	*Process:* EcDsScienceDataServer *Library:* DsCI *Classes:* DsCIRequest, DsCICommand	*Process:* EcDpAtStageDAP *Library:* DpAtDsrv *Class*: DpAtDsrv	The **EcDpAtStageDAP** sends requests to the EcDsScienceDataServer for particular data granules to be pushed, via the FTP service, onto the DPS science processor as input for data processing or for SSIT work.
Insert SSAP Information	One per SSAP	*Process:* EcDsScienceDataServer *Library:* DsCI *Classes:* DsCIRequest, DsCICommand	Operations Staff *Process:* EcDpAtSSAPGui *Library:* DpAtDsrv *Classes:* DpAtSSAPManager, DpAtDsrv	The Operations Staff uses the **EcDpAtSSAPGui** to send requests to the EcDsScienceDataServer to insert new SSAP information or update some existing SSAP information.

Table 4.1-4. SDSRV CSCI Process Interface Events (13 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Export Location Information	One per server	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	*Process:* EcDsScienceDataServer	The EcDsScienceDataServer places physical and logical location information in the **EcCsIdNameServer**.

Table 4.1-4. SDSRV CSCI Process Interface Events (14 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Communications Support	One service per request	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy *Library (Common):* EcUr *Class:* EcUrServerUR *Library:* event *Class:* EcLgErrorMsg *Process:* EcSbSubServer *Library:* EcSbCl *Classes:* EcClEvent, EcClTriggerEventCb, EcClRegisterEventCb *Process:* EcCsEmailParser *Class:* EcCsEmailParser *Process:* EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Process:* EcDsScienceDataServer *Libraries:* DsDe1, DsBtSh *Classes:* DsDeEventCustomizer, DsDeServiceCustomizer, DsBtSbSbrvNotifier	The DCCI CSCI provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry • Request Distribution Media Options from the Configuration Registry (when it first accesses an Earth Science Data Type.)

Table 4.1-4. SDSRV CSCI Process Interface Events (15 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Register Events	One per ESDT installation	*Process:* EcSbSubServer *Library:* EcSbSrSh *Class:* EcSbEvent	*Process:* EcDsScienceDataServer *Library:* DsDe1 *Class:* DsDeEventCustomizer	The EcDsScienceDataServer sends the subscription events for an Earth Science Data Type to the **EcSbSubServer** when an ESDT is installed into the system or when an ESDT is updated by adding additional subscription events.
Replace Events	One per ESDT update	*Process:* EcSbSubServer *Library:* EcSbSrSh *Class:* EcClEvent	*Process:* EcDsScienceDataServer *Library:* DsDe1 *Class:* DsDeEventCustomizer	The EcDsScienceDataServer sends the updated subscription events for an Earth Science Data Type (ESDT) to the **EcSbSubServer** when an ESDT is updated in the system. This replaces the previous information.
Trigger Events	One per subscription event	*Process:* EcSbSubServer *Library:* EcSbCl *Class:* EcClEvent	*Process:* EcDsScienceDataServer *Library:* DsBtSh *Class*: DsBtSbsrvNotifier	The EcDsScienceDataServer notifies the **EcSbSubServer** (via an event trigger) when a subscription event occurs on an Earth Science Data Type Service.
Return URs	One per query	*Process:* EcCsMtMGateway *Classes:* EcCsMtMDataServerMgr, EcCsMtMECSOrderProxy, EcCsMtMOrderImp, EcCsMtMECSSearchOrderProxy, EcCsMtMSearchOrderImp	*Process:* EcDsScienceDataServer *Libraries:* DsCl, DsSh *Classes*: DsClRequest, DsClCommand, DsClESDTReferenceCollector	The EcDsScienceDataServer returns a list of granule URs as a result of a query from the **EcCsMtMGateway**.

Table 4.1-4. SDSRV CSCI Process Interface Events (16 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Inventory Granule Metadata	Once per granule inspect	*Process:* EcCsMtMGateway *Class:* EcCsMtMECSSearchProxy, EcCsMtMSearchImp, EcCsMtMSdsrvMgr, EcCsMtMDataServerMgr	*Process:* EcDsScienceDataServer *Libraries:* DsCl, DsSh *Classes*: DsClESDTReference, DsClESDTReferenceCollector	The EcDsScienceDataServer returns granule metadata information as results of receiving a granule inspect request from the **EcCsMtMGateway**.
Send Acquire	One per acquire	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest DsClCommand DsClESDTReferenceCollector	*Process:* EcCsEmailParser *Class:* EcCsEmailParser *Process:* EcSbSubServer *Library:* EcSbSr *Class:* EcSbSubscription	An “acquire” (instruction to obtain data) is created by the **EcCsEmailParser** or the **EcSbSubServer** and sent to the EcDsScienceDataServer. This is similar to the “Request Product” interface event, except it applies to EDOS expedited data.
Get Info for EDN	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClESDTReference	*Process:* EcCsEmailParser *Class:* EcCsEmailParser	The **EcCsEmailParser** sends requests to the EcDsScienceDataServer for the Expedited Data Set Notification (EDN) information and sends messages to users at the ASTER GDS.
Send Search	Per client request	*Processes:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClESDTReferenceCollector	*Process:* EcCsMtMGateway *Library:* DsXSsStaticClientLibs (macro) *Classes:* See note at the end of this table.	The **EcCsMtMGateway** sends inventory search requests, constructed from the qualifying metadata information in the SIPS request, to the EcDsScienceDataServer via a remote procedure call.
Send Event ID	One per event	*Process:* EcDsScienceDataServer *Library:* DsDe1 *Class:* DsDeEventCustomizer	*Process:* EcSbSubServer *Library:* EcSbSrSh *Class:* EcSbEvent	The **EcSbSubServer** sends Event IDs to the EcDsScienceDataServer when ESDTs are installed or when ESDTs are updated by adding additional events.

Table 4.1-4. SDSRV CSCI Process Interface Events (17 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Status	One per request	*Process:* EcDsScienceDataServer *Library:* DsBtSh *Class:* DsBtSbsrvNotifier	*Process:* EcSbSubServer *Library:* EcUt	Status returned by the **EcSbSubServer** to the EcDsScienceDataServer to simply indicate that the request was received, not that the action succeeded.
Import Location Information	One per server	*Process:* EcDsScienceDataServer	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	The EcDsScienceDataServer retrieves server location information from the **EcCsIdNameServer**.
Return Configuration Parameters	One set per request	*Process:* EcDsScienceDataServer	*Process:* EcCsRegistry *Library:* EcCsRegistry Class: EcRgRegistryServer_C	The **EcCsRegistry** returns the attribute-value pairs (configuration parameters) to the EcDsScienceDataServer upon request.
Return Dist. Media Options	One set of media types per request	*Process:* EcDsScienceDataServer *Library:* DsDe1 *Class:* DsDeServiceCustomizer	*Process:* EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	The **EcCsRegistry** returns the attribute-value pairs (configuration parameters) to the EcDsScienceDataServer upon request.
Request Granule Deletion	One per granule delete request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest DsClCommand DsClESDTReferenceCollector	*Process:* EcDpPrDeletion *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The **EcDpPrDeletion** sends delete requests to the EcDsScienceDataServer for particular granules (interim data) in the SDSRV archives.

Table 4.1-4. SDSRV CSCI Process Interface Events (18 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Data InsertPr	One per data insert request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsGeESDT	*Processes:* EcDpPrEM, EcDpPrQaMonitorGUI *Libraries:* DpPrDssIF, DpPrQaMonitor *Classes:* DpPrDSSInterface, DpPrQAGranuleQaFlags	The **EcDpPrEM** and **EcDpPrQaMonitorGUI** processes send requests to the EcDsScienceDataServer to insert a particular file or files into the archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. For the EcDpPrEM process, these files can be granules or PGE tar files. For the EcDpPrQaMonitorGUI, these files are metadata updates.
Acquire PGE Tar File	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The **EcDpPrEM** process acquires a tar file for any PGE not currently local to the science processor from the EcDsScienceDataServer. The executable is extracted from the tar file and used during PGE execution.
Insert Product History Tar Files	One per successful PGE execution	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	After the PGE has successfully completed executing and archiving the resulting outputs, the **EcDpPrEM** requests the PGE Production History Tar file be inserted into the EcDsScienceDataServer for permanent archive.
Insert Failed PGE Tar File	One per unsuccessful PGE execution	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	After an unsuccessful execution of a PGE, the **EcDpPrEM** obtains the Tar file containing the PGE log files, core dump (if any), PCF and other files, and requests the files be inserted into the EcDsScienceDataServer for permanent archive.

Table 4.1-4. SDSRV CSCI Process Interface Events (19 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Update Product Metadata	One per metadata product update	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClCommand, DsClRequest, DsClESDTReferenceCollector	*Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQAGranuleQaFlags	The **EcDpPrQaMonitorGUI** provides the operator with capabilities to update product metadata in the EcDsScienceDataServer.
Retrieve Product History	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClAcquireCommand	*Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQaMonitor	The **EcDpPrQaMonitorGUI** submits requests of this type to the EcDsScienceDataServer. It transfers the Production History tar file from the Science Data archive to the user's host machine.
Query Data	One per query	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClESDTReferenceCollector	Operations Staff *Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQaDataGranule	The Operations Staff uses the **EcDpPrQaMonitorGUI** to submit requests of this type to the EcDsScienceDataServer. It searches the archive for granules that match the user-supplied selection criteria: data type and begin/end date. Results are displayed to the user at the EcDpPrQaMonitorGUI.
Retrieve Data	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClQuery, DsClAcquireCommand	Operations Staff *Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQaMonitor	The Operations Staff uses the **EcDpPrQaMonitorGUI** to send retrieval requests, to the EcDsScienceDataServer, for a particular data granuleId. The product is transferred (pushed), via the FTP service, onto the DPS science processor and used as input for PGE processing or for SSIT work.

Table 4.1-4. SDSRV CSCI Process Interface Events (20 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Return MCF Template	One per set of external data received by EMD	*Processes:* EcDpPrEM, EcDpAtGetMCF *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	The EcDsScienceDataServer provides the MCF template as part of the GetMCF service call to the **EcDpPrEM** and **EcDpAtGetMCF** processes.
Return PGE Tar File	One per request	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	After an unsuccessful execution of a PGE, the **EcDpPrEM** obtains the Tar file containing the PGE log files, core dump (if any), Process Control File (PCF) and other files, and requests the files be inserted into the DSS for permanent archive.
Search Inventory	One per service request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClQuery, DsClESDTReferenceCollector	*Process:* EcDmV0ToGateway *Library:* RequestProcessing *Class:* DmGwInventoryRequest	The **EcDmV0ToEcsGateway** sends requests to the EcDsScienceDataServer to search the SDPS Inventory (archives).
Request Browse Data	One per browse request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClRequest, DsClCommand	*Process:* EcDmV0ToEcsGateway *Library:* RequestProcessing *Class*: DmGwBrowseRequest	The **EcDmV0ToEcsGateway** submits requests for browse data to the EcDsScienceDataServer to acquire reduced resolution products to support a product request.
Request Product	One per product order request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDmV0ToEcsGateway *Library:* RequestProcessing *Class:* DmGwAcquireRequest	The **EcDmV0ToEcsGateway** forwards product requests to the EcDsScienceDataServer from an external user to be distributed by the EcDsDistributionServer upon receipt of the data from the EcDsStStagingDiskServer.

Table 4.1-4. SDSRV CSCI Process Interface Events (21 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Inspect	One per price estimate request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClRequest	*Process:* EcDmV0ToEcsGateway *Library:* RequestProcessing *Class:* DmGwPriceEstimateRequest	The **EcDmV0ToEcsGateway** sends a request for an inspection of granule metadata to the EcDsScienceDataServer in support of a price estimate request.
Return Inventory Granule Metadata	One per service request	*Process:* EcDmV0ToEcsGateway *Library:* RequestProcessing *Class:* DmGwInventoryRequest	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClQuery, DsClESDTReferenceCollector	The EcDsScienceDataServer returns ESDT Universal References (URs) for the requested granules to the **EcDmV0ToEcsGateway**.
Return Granule URs	One per price estimate request	*Process:* EcDmV0ToEcsGateway *Library:* RequestProcessing *Class:* DmGwPriceEstimateRequest	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClRequest	The EcDsScienceDataServer returns the URs for the granules to the **EcDmV0ToEcsGateway** to inspect the metadata in support of a price estimate request.
Return Browse Data	One per browse request	*Process:* EcDmV0ToEcsGateway *Library:* RequestProcessing *Class*: DmGwBrowseRequest	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClRequest, DsClCommand	The EcDsScienceDataServer returns browse data to the **EcDmV0ToEcsGateway** to acquire reduced resolution products to support a product request.
Return Product	One per product order request	*Process:* EcDmV0ToEcsGateway *Library:* RequestProcessing *Class:* DmGwAcquireRequest	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	The EcDsScienceDataServer returns products to the **EcDmV0ToEcsGateway** to forward to an external user via the EcDsDistributionServer upon receipt of the data from the EcDsStStagingDiskServer.

Table 4.1-4. SDSRV CSCI Process Interface Events (22 of 22)

Event	Event Frequency	Interface	Initiated By	Event Description
Return SSAP Information	One per SSAP information request	*Process:* EcDpAtSSAPGui *Libraries:* DpAtSSAP, DpAtDsrv *Classes:* DpAtSSAPManager, DpAtDsrv	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClRequest, DsClCommand	The EcDsScienceDataServer returns information about SSAPs, including names of existing SSAPs and the components associated with a specific SSAP to the **EcDpAtSSAPGui**.
Return DAPs	One DAP per request	*Process:* EcDpAtStageDAP *Library:* DpAtDsrv *Class:* DpAtDsrv	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClRequest, DsClCommand	The EcDsScienceDataServer returns DAPs from the SDPS archives to the **EcDpAtStageDAP,** which stores the DAPs on the local AITTL disk.
Send ESDT Installation Information	One per new ESDT installation	*Process:* EcDmDictServer *Library:* EcDmDdClient *Class:* DmDdClSchemaRequest	*Process*: EcDsScienceDataServer *Library:* DsDe1 *Class:* DsDeDataDictController	The EcDsScienceDataServer sends ESDT installation information, to the **EcDmDictServer**, whenever a new ESDT is installed. This data consists of Inventory and Collection level metadata.
Replace ESDT Installation Information	One per ESDT update	*Process:* EcDmDictServer *Library:* EcDmDdClient *Class:* DmDdClSchemaRequest	*Process:* EcDsScienceDataServer *Library:* DsDe1 *Class:* DsDeDataDictController	The EcDsScienceDataServer sends updated ESDT information to the **EcDmDictServer** whenever an ESDT is updated. This data consists of updated Inventory and Collection level metadata.
Get File Sizes				The **EcPdPDS** sends a request to the EcDsScienceDataServer to get file sizes.
Submit Acquire				The **EcPdPDS** submits an acquire request to the EcDsScienceDataServer to retrieve data granules of interest.

Note: DsXSsStaticClientLibs is a macro defined in the COMMON components. It defines DSS library includes for the Cl, Sh, Ge, Sr, De2 and Gl libraries. The important classes used from this macro (by library include) are:

- **Cl** – DsClCommand, DsClDescriptor, DsClESDTAddRequest, DsClESDTReference, DsClESDTReferenceCollector, DsClESDTReferenceVector, DsClInsertCommand, DsClQuery and DsClRequest
- **Sh** – DsShByteBuffer, DsShConnectionProxy, DsSh DescriptorProxy, DsShESDTUR, DsShError, DsShErrorDetails, DsShMutex, DsShMutexLock, DsShRequest, DsShRequestProxy and DsShThread
- **Gl** – GlParameterList, GlParameter, GlLongP, GlGPolygonP, GlDoubleP, GlBinaryP, GlRectangleP, GlStringP and GlTimeP

4.1.1.6 Science Data Server Data Stores

Table 4.1-6 provides a description of the data stores for the SDSRV CSCI, and the conceptual model of the data store. The physical model for the SDSRV data stores can be found in the Science Data Server Database Design and Schema Specifications for the EMD Project (CDRL 311).

Table 4.1-5. SDSRV CSCI Data Stores

Data Store	Type	Description
EMD Inventory	Database	The EMD Inventory (archives) contains the metadata describing the earth science data for the Earth Science Data Types at a specific DAAC. The metadata describes: • Collection level information • Browse data • Science data (as granules) • Quality Assessments • Algorithm Packages • Delivered Algorithm Packages • Production History The EMD catalog also contains systems data for the dynamic configuration of the EcDsScienceDataServer. The EMD catalog also contains implementation of the “EMD Data Model” for Attribute Valids checking. The EMD catalog also contains system data for ESDT Configuration.

4.1.2 Data Distribution Software Description

4.1.2.1 Data Distribution Functional Overview

The Data Distribution (DDIST) CSCI monitors and controls processing for distribution requests. Data Distribution processing consists of directing the STMGT CSCI to place data for distribution in working storage, creating packing lists, directing the STMGT CSCI to "FtpPush", "FtpPull", or "SecureCopy", and sending notifications for completed distribution requests. The DDIST CSCI alone or through the use of the STMGT CSCI does not place data on hard media. For FtpPush requests, the STMGT FTP Server (when requested by the Distribution Server) pushes data to a user-specified location. For FtpPull requests, data is placed in a directory to be pulled. For SecureCopy, the STMGT CopyServer copies the data to a user-specified directory. Once the data is ready, the DDIST CSCI sends an electronic message (via eMail or scp) to the user providing the required information for the user to pull the data. The DDIST CSCI has a GUI interface with the administration/operations staff (Admin. /Ops). The GUI provides error conditions and status to operations staff and enables the operations staff to set parameters and control operations including suspending, canceling, and resuming requests, changing the priorities of requests, performing multiple selects, and setting threshold sizes. The DDIST CSCI provides limited automatic error response by suspending requests when most errors are encountered.

The DDIST CSCI has an interface with the following:

- STMGT CSCI
- SDSRV CSCI
- Administrator/Operations staff (through a GUI or command line)
- MSS (MCI)

The Administrator/Operations staff and the SDSRV CSCI control the DDIST CSCI activities. The DDIST CSCI receives direction to perform its functions from calls to the STMGT CSCI.

4.1.2.2 Data Distribution Context

Figure 4.1.2-1 is the DDIST CSCI context diagrams. The diagrams show the events sent to the DDIST CSCI and the events the DDIST CSCI sends to other CSCIs and Users. Table 4.1.2-1 provides descriptions of the events (by service name) shown in the DDIST CSCI context diagrams.

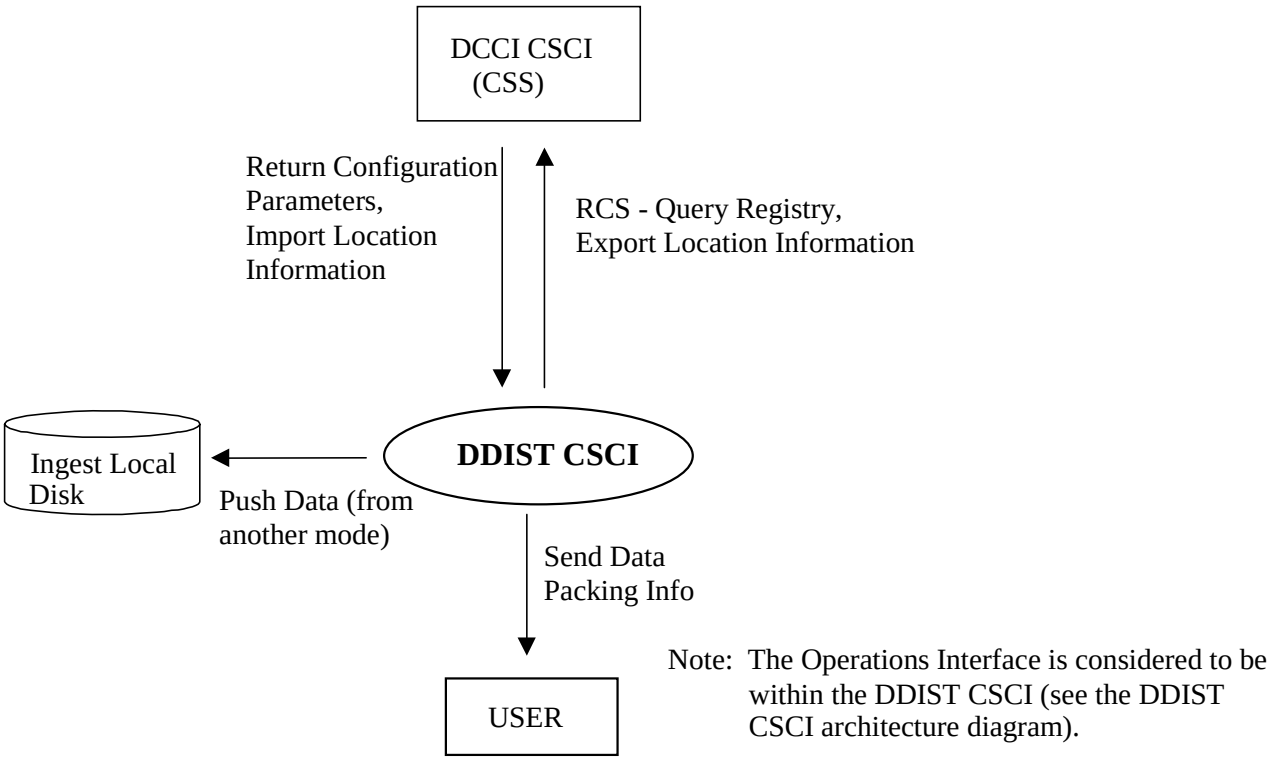


Figure 4.1.2-1. DDIST CSCI Context Diagram

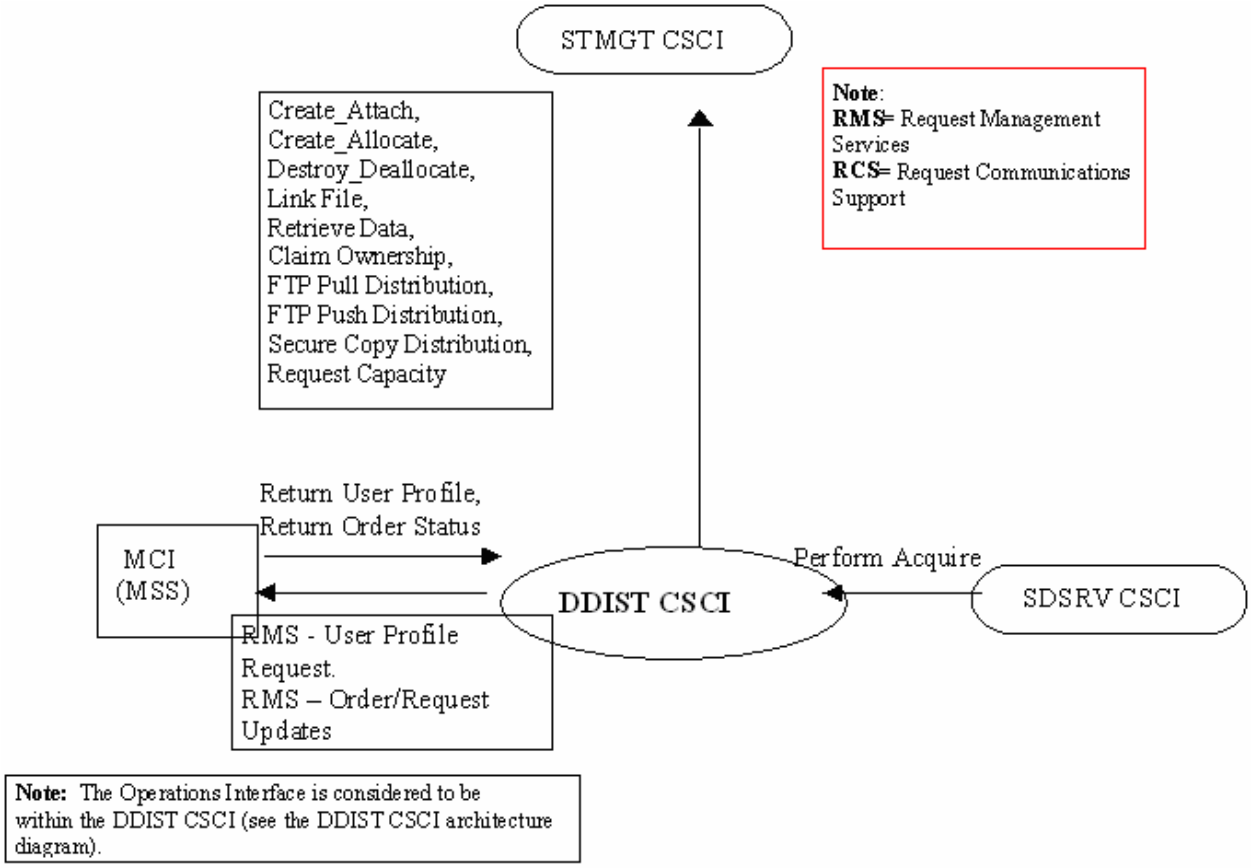


Figure 4.1.2-1. DDIST CSCI Context Diagram (cont.)

Table 4.1.2-1. DDIST CSCI Interface Events (1 of 2)

Event	Interface Event Description
Request Communications Support	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Export Location Information	The DDIST CSCI stores physical and logical server location information in the **DCCI CSCI**.
Send Data Packing Info	Notification sent to the **user** via E-mail about a data product to be or has been distributed. Data handled electronically is either pushed, via the File Transfer Protocol (FTP) service, to a user-specified location or placed in a directory to be pulled via the FTP service by the user.
Push Data (from another mode)	The DDIST CSCI pushes data, via the FTP service, to the **Ingest local disk** when it is distributing data to be ingested.
Return Configuration Parameters	The DDIST CSCI receives the configuration parameters and associated values from the Registry Server within the **DCCI CSCI**.
Import Location Information	The DDIST CSCI retrieves physical and logical server location information from the **DCCI CSCI**.
Create_Attach	The DDIST CSCI sends requests to the **STMGT CSCI** to attach (gain access to an existing staging disk area allocated by another process) to a staging disk area.
Create_Allocate	The DDIST CSCI sends requests to the **STMGT CSCI** to allocate areas on the local staging disk to store data for distribution.
Destroy_Deallocate	The DDIST CSCI sends requests to the **STMGT CSCI** to deallocate (lose access to) an existing staging disk area.
Link File	The DDIST CSCI sends requests to the **STMGT CSCI** to link files from read-only cache to a staging disk specified in the request.
Retrieve Data	The DDIST CSCI sends requests to the **STMGT CSCI** to retrieve data from the SDPS archives to be staged for distribution.
Claim Ownership	The DDIST CSCI sends requests to the **STMGT CSCI** to claim ownership of (take responsibility for deallocating) an existing staging disk area.
FTP Pull Distribution	The DDIST CSCI sends requests to the **STMGT CSCI** to move a file to the Pull area.
FTP Push Distribution	The DDIST CSCI sends requests to the **STMGT CSCI** to distribute a file directly to a user or to a predetermined local disk area for the user.
Secure Copy Distribution	The DDIST CSCI sends requests to the **STMGT CSCI** to distribute a file to a user specified disk area using Secure Copy.

Table 4.1.2-1. DDIST CSCI Interface Events (2 of 2)

Event	Interface Event Description
Request Capacity	The DDIST CSCI sends requests to the **STMGT CSCI** to get the effective maximum capacity of a media type. This information is used to determine the number of media needed to satisfy a given request.
Perform Acquire	A request is sent from the **SDSRV CSCI** to the DDIST CSCI for science data or a product to be sent to a specified user. The **SDSRV CSCI** assembles instructions to send data and sends the instructions to the DDIST CSCI. The DDIST CSCI stores the request received from the **SDSRV CSCI** in a queue with the appropriate priority. The DDIST CSCI sends a request to the STMGT CSCI to push the data, via the FTP service. When the request is taken from the queue, the STMGT CSCI is passed the data retrieval request from the DDIST CSCI for a particular data granule to be pushed onto the DPS science processor, via the FTP service. The data granule is to be used as input for PGE processing or for SSIT work.
Send Distribution Notification	The DDIST CSCI sends a distribution notification, via e-mail, to the **PDSIS CSCI** when data is being distributed on hard media.
Check Distribution Status	The **PDSIS CSCI** checks the data base table for the status of the distribution request (i.e., in the queue, being processed, or distribution completed).
Request Management Services	The **MCI** provides a basic management library of services to the CSCIs, implemented as client or server applications, using the CSS Process Framework. The basic management library of services includes: • **System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document. The MCI also interfaces with other CSCIs or CSCs to perform the following: • **User Profile Request** – The MCI provides requesting CSCIs or CSCs with access to user profile information such as e-mail address and shipping address to support their processing activities. • **Order/Request Updates** - The DDIST CSCI interfaces with the Accountability Management Service Order/Request Tracking to create/update the EcAcRequest (user product order request) such as media id, quantity and type.
Return User Profile	The DDIST CSCI receives user profile information from the **MCI** to authenticate a user.
Return Order Status	The DDIST CSCI receives order status (i.e., Order ID, Request ID) information from the **MCI**.

4.1.2.3 Data Distribution Architecture

The DDIST CSCI is a couple of SDPS developed processes with the addition of the Sybase ASE COTS hardware and software process package as a data repository identified as:

- EcDsDistributionServer - Data Distribution
- EcDsDdistGui - Data Distribution GUI
- Sybase ASE - Data Repository (storage area)

Figure 4.1.2-2 is the DDIST CSCI architecture diagrams. The diagrams show the events sent to the DDIST CSCI processes and the events the DDIST CSCI processes send to other processes and the Operations staff.

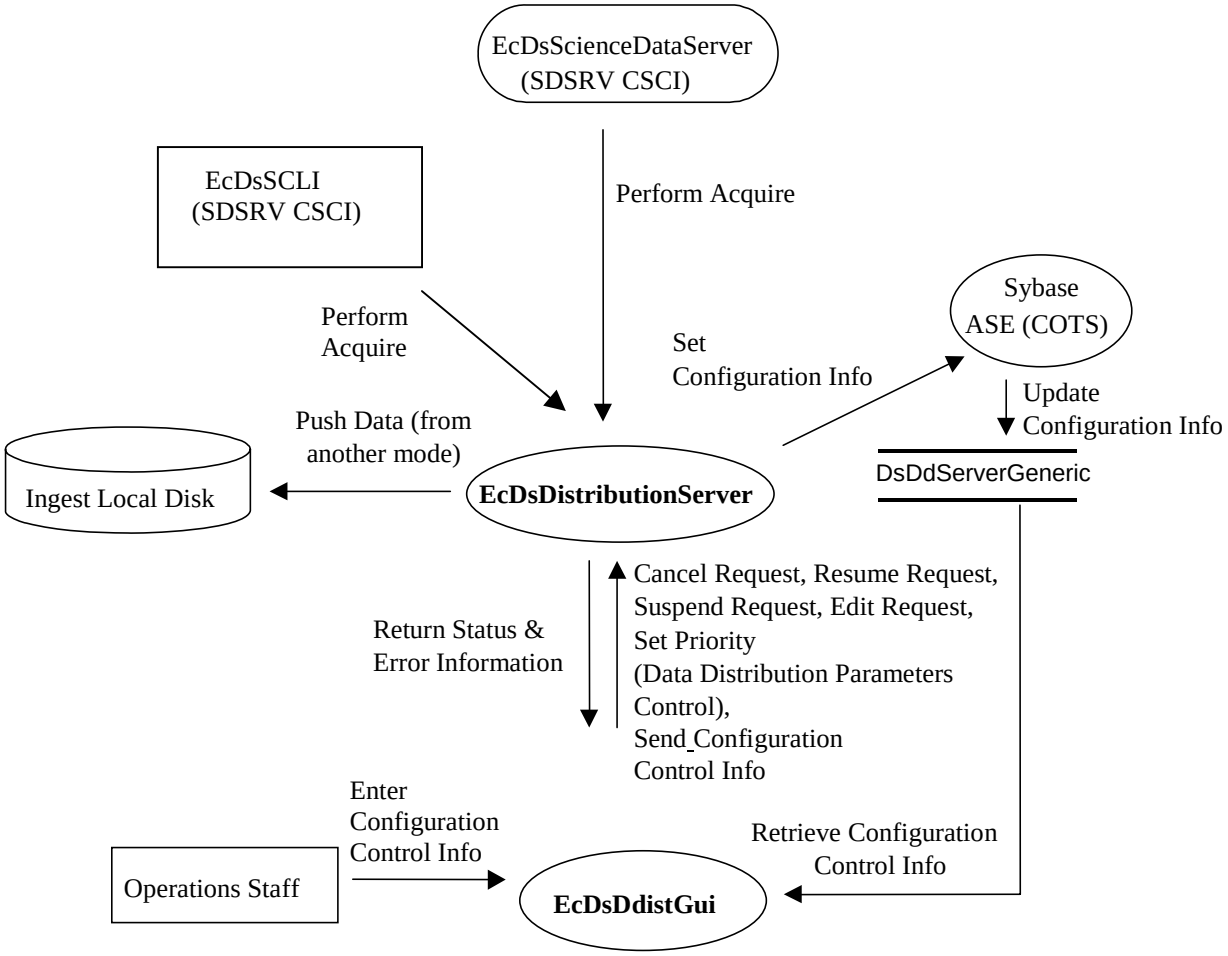


Figure 4.1.2-2. DDIST CSCI Architecture Diagram

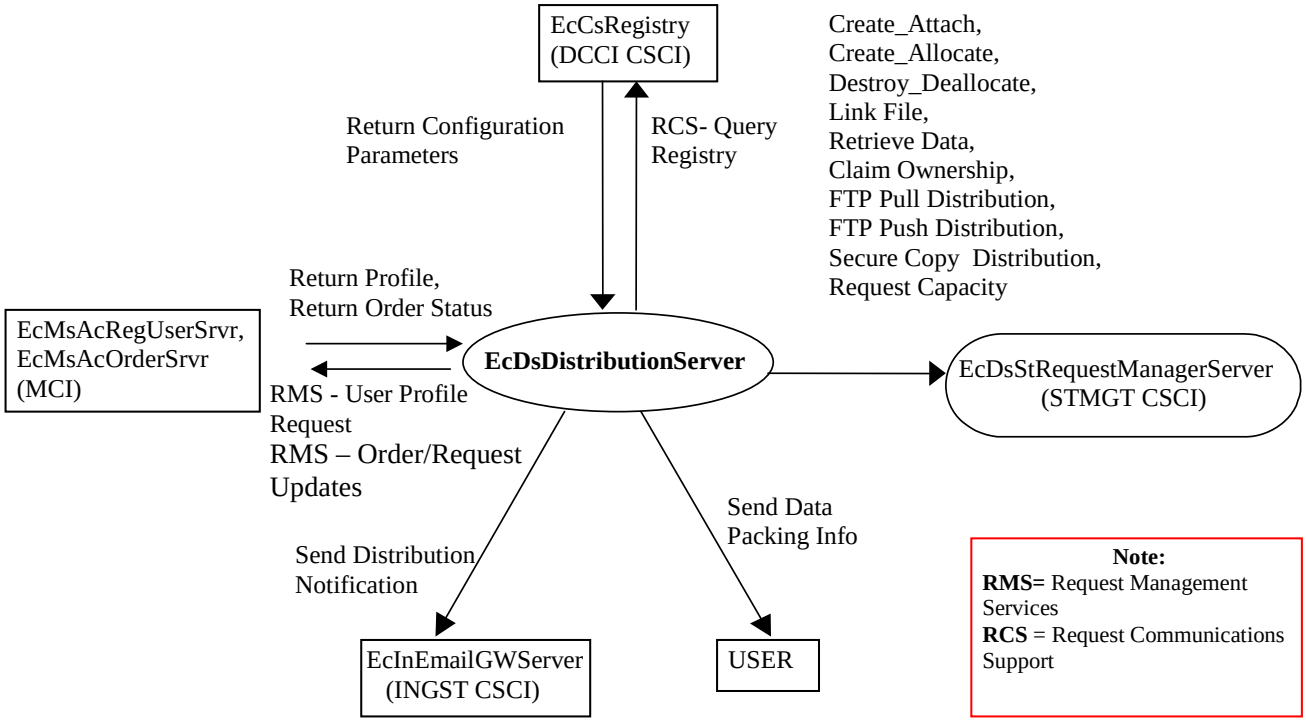


Figure 4.1.2-2. DDIST CSCI Architecture Diagram (cont.)

4.1.2.4 Data Distribution Process Descriptions

Table 4.1.2-2 provides descriptions of the processes shown in the DDIST CSCI architecture diagram.

Table 4.1.2-2. DDIST CSCI Processes

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcDsDistributionServer	Server	ACMHW	Developed	This process provides the control and coordination for data distribution through request processing.
EcDsDdistGui	GUI	ACMHW	Developed	This process enables operations to initiate, track, and manipulate distribution requests by using input GUI controls and database information.
Sybase ASE	Server	ACMHW	COTS	The process contains the request list and has a set of stored procedures, which updates the request configuration, provides the request configuration to GUI operations and checkpoints the state of the CSCI for fault recovery purposes.

In the EMD Baseline Information System (EBIS) Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.1.2.5 Data Distribution Process Interface Descriptions

Table 4.1.2-3 provides descriptions of the interface events shown in the DDIST CSCI architecture diagrams.

Table 4.1.2-3. DDIST CSCI Process Interface Events (1 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
Perform Acquire	One granule and file per request	*Process:* EcDsDistributionServer *Libraries:* DsDdSSh, DsDdB, DsDdC, DsDdl *Classes:* DsDdScheduler, DsDdRequestMgrReal, DsDdDCERequestMgrConcrete, DsDdRequestMgrC, DsDdGranuleC, DsDdGranuleS	*Process:* EcDsScienceDataServer *Library*: DsSr *Classes:* DsDbInterface, DsDdRequestMgrReal, DsSrWorkingCollection	The **EcDsScienceDataServer** sends requests to the EcDsDistributionServer for science data (granule(s)) or a product to be sent to a specified user. The **EcDsScienceDataServer** assembles instructions to send data to the EcDsDistributionServer. The EcDsDistributionServer sends a request to the EcDsStFtpServerServer to push the data, via the FTP service, followed by a signal file to the destination specified in an acquire instruction by particular ESDTs that function this way. The **EcDsScienceDataServer** sends requests to the EcDsDistributionServer to retrieve subsetted data files for distribution.

Table 4.1.2-3. DDIST CSCI Process Interface Events (2 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
Set Configuration Info	One per scheduling request	Sybase ASE (COTS)	*Process:* EcDsDistributionServer *Libraries:* DsDdSSh, DsDdC *Classes:* DsDdRequestMgrBaseC, DsDdConfiguration, DsDdDistRequestS	Current configuration information entered by the Operations staff via the EcDsDdistGui is sent to the **Sybase ASE** database via the EcDsDistributionServer. The Operations staff can access the configuration information from the database for expedient data distribution or product order distribution scheduling.
Update Configuration Info	One set per request	DDIST Data Stores	*Process:* Sybase ASE (COTS*)*	The **Sybase ASE** updates the configuration data in the data stores as requested.
Retrieve Configuration Control Info	Upon Operations Staff request	DDIST Data Stores	*Process:* EcDsDdistGui *Classes:* DsDdConfiguration, DsDdPfConfigFile	The Operations Staff retrieve configuration control information from the database for viewing or update via the **EcDsDdistGui**.
Cancel Request	One per priority cancel request	*Process:* EcDsDistributionServer *Library:* DsDdC *Class:* DsDdRequestMgrBaseC	Operations Staff *Process:* EcDsDdistGui *Class:* DsGuiDistRequest	The Operations Staff uses the **EcDsDdistGui** to send a command to the EcDsDistributionServer to cancel a request.
Resume Request	One per resume request	*Process:* EcDsDistributionServer *Libraries:* DsDdB, DsDdC, DsDdI *Class:* DsDdRequestMgrC	Operations Staff *Process:* EcDsDdistGui *Class:* DsGuiDistRequest	The Operations Staff uses the **EcDsDdistGui** to send a command to the EcDsDistributionServer to resume requests when requests are suspended with errors or suspended.
Suspend Request	One per suspend request	*Process:* EcDsDistributionServer *Libraries:* DsDdB, DsDdC, DsDdI *Class:* DsDdRequestMgrC	Operations staff *Process:* EcDsDdistGui *Class:* DsGuiDistRequest	The **Operations staff** uses the EcDsDdistGui to send a command to the EcDsDistributionServer to suspend a request(s).

Table 4.1.2-3. DDIST CSCI Process Interface Events (3 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
Edit Request	Per operator request on any given suspended request	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Classes:* DsDdScheduler, DsDdRequestMgrReal, DsDdDCERequestMgrConcrete	Operations Staff *Process:* EcDsDdistGui *Class:* DsGuiDistRequest	The **Operations staff** uses the EcDsDdistGui to send requests to the EcDsDistributionServer to edit a suspended request to the selected media type.
Set Priority	One per priority change	*Process:* EcDsDistributionServer *Library:* DsDdS *Classes:* DsDdMedia, DsDdRequestListS, DsDdRequestMgrReal, DsDdDistRequestS	Operations staff *Process:* EcDsDdistGui *Class:* DsGuiDistRequest	The **Operations staff** uses the EcDsDdistGui to send a command to the EcDsDistributionServer to change the priority of a distribution request.
Send Configuration Control Info	One per request	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Classes:* DsDdDistListS, DsDdDistFileS, DsDdDistRequestS	*Process:* EcDsDdistGui *Classes:* DsDdConfiguration DsDdPfConfigFile	The configuration control information captured by the EcDsDdistGui is sent to the EcDsDistributionServer to be stored in the DDIST data stores.
Enter Configuration Control Info	One per configuration sent	*Process:* EcDsDdistGui *Classes:* DsDdConfiguration DsDdPfConfigFile	Operations staff	The **Operations staff** enters configuration control information (queues, thresholds, suspend and resume status) via the EcDsDdistGui or command line.
Return Status & Error Information	One per Distribution Request	*Process:* EcDsDdistGui *Class:* DsGuErrorDialog	*Process:* EcDsDistributionServer *Classes:* DsDdMedia, DsDdBaseQueue	The EcDsDistributionServer sends Distribution Request status and error information to the Operations staff via the **EcDsDdistGui**, if the acquire is asynchronous and the EcDsDistributionServer gets the request.

Table 4.1.2-3. DDIST CSCI Process Interface Events (4 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
Push Data (from another mode)	One distribution per request	Ingest Local Disk	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	The EcDsDistributionServer pushes data, via the FTP Service, to the **Ingest Local Disk** when it is distributing data to be ingested.
Request Communications Support	One service per request	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Library:*** event *Class:* EcLgErrorMsg ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Processes:* EcDsDistributionServer *Classes:* DsDdRequestMgrServer, DsDdlog	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Create_Attach	One attach per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Classes:* DsDdMedia, DsDsStaging	The EcDsDistributionServer sends requests to the **EcDsStRequestManagerServer** to attach (gain access to an existing staging disk area allocated by another process) to a staging disk area.

Table 4.1.2-3. DDIST CSCI Process Interface Events (5 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
Create_Allocate	One allocation per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdMedia	The EcDsDistributionServer send requests to the **EcDsStRequestManagerServer** to allocate areas on the local staging disk to store data for distribution.
Destroy_Deallocate	One deallocation per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdMedia	The EcDsDistributionServer sends requests to the **EcDsStRequestManagerServer** to deallocate (lose access to) an existing staging disk area.
Link File	One link per file in a request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Library:* DsDdS *Class:* DsDdGranuleS	The EcDsDistributionServer sends requests to the **EcDsStRequestManagerServer** to link files from the read-only cache to a staging disk and from one staging disk to another.
Retrieve Data	One granule per retrieval request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	The EcDsDistributionServer sends requests to the **EcDsStRequestManagerServer** to retrieve data from the SDPS archives.
Claim Ownership	One per staging disk area	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdMedia	The EcDsDistributionServer sends requests to the **EcDsStRequestManagerServer** to claim ownership of (take responsibility for deallocating) an existing staging disk area.

Table 4.1.2-3. DDIST CSCI Process Interface Events (6 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
FTP Pull Distribution	One order per pull request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	EcDsDistributionServer sends requests to the **EcDsStRequestManagerServer** to move a file to the Pull area.
FTP Push Distribution	One order per push request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	EcDsDistributionServer sends requests to the **EcDsStRequestManagerServer** to distribute a file directly to a user via a local disk area accessible by the user.
Secure Copy Distribution	One order per scp request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	EcDsDistributionServer sends a request through a socket connection to the EcDsStCopyServer to scp a file to a user specified directory.
Request Capacity	One calculation per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdMedia	EcDsDistributionServer determines the effective maximum capacity of a media type to send to the **EcDsStRequestManagerServer**. This is used to determine the number of media needed to satisfy a given request.
Send Data Packing Info	One per distributed data location	User	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Class:* DsDdMedia	The EcDsDistributionServer sends notifications to the **user** via e-mail or scp with the location of data to be distributed or has been distributed.
Send Distribution Notification	One per distribution	*Process:* EcInEmailGWServer *Classes:* InEmailGWServer, InEmailParser	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Classes:* DsDdMedia, DsDdMediaDist	The EcDsDistributionServer sends distribution notifications, via e-mail, to the **EcInEmailGWServer**.

Table 4.1.2-3. DDIST CSCI Process Interface Events (7 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Management Services (RMS)	At system startup or shutdown and for restarts	*Process:* EcDsDistributionServer	DAAC unique startup scripts	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document.
RMS (cont.)	One per address request	*Process:* EcMsAcRegUserSrvr *Libraries:* MsAcClnt, MsAcComm	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Class:* DsDdMedia	The **MCI** also interfaces with other CSCIs or CSCs to perform the following: **User Profile Request** – The MCI provides requesting CSCIs or CSCs with access to user profile information such as e-mail address and shipping address to support their processing activities.
RMS (cont.)	One per address request	*Processes:* EcMsAcOrderSrvr *Libraries:* MsAcClnt, MsAcComm	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Class:* DsDdMedia	**Order/Request Updates** - The DDIST CSCI interfaces with the **MCI** Accountability Service Order/Request Tracking to create/update a user product order request.
Return User Profile	One profile per request	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Class:* DsDdMedia	*Process:* EcMsAcRegUserSrvr *Libraries:* MsAcClnt, MsAcComm *Class:* MsAcUsrProfile	The **EcMsAcRegUserSrvr** returns user profile information requested by the EcDsDistributionServer.

Table 4.1.2-3. DDIST CSCI Process Interface Events (8 of 8)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Order Status	One per order request	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Classes:* DsDdMedia, DsDdRequestMgrServer	*Process:* EcMsAcOrderSrvr *Library:* MsAcClnt *Class:* EcAcOrderCMgr	The **EcMsAcOrderSrvr** provides order status information (i.e., Order ID, Request ID) for products requested by the EcDsDistributionServer.
Return Configuration Parameters	One set per request	*Process:* EcDsDistributionServer *Library:* EcCsRegistry Class: EcRgRegistryServer_C	*Process:* EcCsRegistry *Library:* EcCsRegistry Class: EcRgRegistryServer_C	The **EcCsRegistry** returns the attribute-value pairs (configuration parameters) to the EcDsDistributionServer upon request.

4.1.2.6 Data Distribution Data Stores

Table 4.1.2-4 provides descriptions of the individual DDIST CSCI data stores entitled collectively "DDIST Data Stores" in the DDIST CSCI architecture diagram. More details on these database tables can be found in the Data Distribution Database Design and Schema Specifications for the EMD Project.

Table 4.1.2-4. DDIST CSCI Data Stores (1 of 2)

Data Store	Type	Description
DsDdFile	Sybase	This data store holds the distribution files maintained/processed by the EcDsDistributionServer. Table Abbreviation "F" is used as the standard naming convention for stored procedures.
DsDdGranule	Sybase	This data store holds the distribution granule maintained/processed by the EcDsDistributionServer. Table Abbreviation "G" is used as the standard naming convention for stored procedures.
DsDdParameterList	Sybase	This data store holds the GlParameter list for each request maintained/processed by the EcDsDistributionServer. The EcDsScienceDataServer provides data from external metadata (i.e., via the MCF). Request information is initiated here first. Table abbreviation "PL" is used as the standard naming convention for stored procedures.

Table 4.1.2-4. DDIST CSCI Data Stores (2 of 2)

Data Store	Type	Description
DsDdRequest	Sybase	This data store holds the distribution requests maintained/processed by the EcDsDistributionServer. Table abbreviation "R" is used as the standard naming convention for stored procedures.
DsDdServerGeneric	Sybase	This data store holds generic configuration settings for the EcDsDistributionServer.
DsDdParameterListArchive	Sybase	This data store holds all versions of the GL Parameter list for each request currently being maintained and processed by the EcDsDistributionServer. Each time a request is edited, a new row is added to this table.
DsDdRequestArchive	Sybase	This data store holds all versions of the distribution requests currently being maintained and processed by the EcDsDistributionServer. Each time a request is edited, a new row is added to the data store.

4.1.3 Storage Management Software Description

4.1.3.1 Storage Management Functional Overview

The Storage Management (STMGT) CSCI stores/archives, manages, and retrieves non-document earth science data and provides a user-friendly graphical user interface (GUI) for operations. The STMGT CSCI manages all physical storage resources for all the DSS CSCIs and processes including: tape robotic archive, RAID disk cache, on-line storage, and peripheral devices used for ingesting data from and distributing data to hard media such as various hard media sizes or drive types.

The STMGT CSCI manages both long-term, high capacity archival of data (data repository) and short term/temporary storage (working storage/cache management). The STMGT CSCI controls associated file access services to the archive, handles short-term data storage needs for the INGST CSCI, the DDIST CSCI, the SDSRV CSCI, and the PRONG CSCI. The STMGT CSCI also provides access to hard media peripheral devices for both the INGST CSCI and the DDIST CSCI.

During data ingest, the STMGT CSCI provides interfaces to enable ingest and obtain access to disk space, FTP services, and shared resource peripheral devices. The STMGT CSCI copies files into the archive for permanent storage. During data distribution, the SDSRV CSCI and the DDIST CSCI copy files from the archive and allocates magnetic disk space for staging the files. The DDIST CSCI also allocates the peripheral devices shared with the INGST CSCI for copying of files to hard media, or to copy files for electronic distribution. The STMGT CSCI maintains a user pull area to allow for electronic pull distribution.

The STMGT CSCI provides retrieval and storage methods to the DDIST CSCI and the SDSRV CSCI to support storing and providing data for their client CSCIs. The PRONG CSCI is a client CSCI in both storage and retrieval requests by retrieving lower level data via FTP Push Acquire

through the SDSRV and DDIST CSCIs and by storing data via insert request to the SDSRV CSCI. In addition to the product files created by the PRONG CSCI, the following types of files are stored for the PRONG CSCI: intermediate product files (interim files), production history files, metadata files, and lower level data files such as raw science data. The STMGT CSCI stores files as a result of insert requests to the SDSRV CSCI by the INGST CSCI and the PRONG CSCI. The STMGT CSCI retrieves files from the archive to satisfy sub-setting requests submitted to the SDSRV CSCI and by acquire requests submitted to the SDSRV CSCI and routed through the DDIST CSCI.

The STMGT GUI provides a simple and consistent interface to set various system parameters, identify errors, analyze the underlying problem(s), and develop corrective measures. The persistence of the data is maintained in a database through a STMGT GUI interface. The GUI provides a method to manage system resources such as servers, cache thresholds, on-line storage availability, and peripherals. The GUI also provides the operator with the capability to track the status of files, hard media, and drives for an enhanced level of quality control.

4.1.3.2 Storage Management Context

Figure 4.1.3-1 is the STMGT CSCI context diagram. The diagram shows the events sent to other CSCIs and events received from other CSCIs.

Note: any items italicized and inside of < > are items which show consistency across the diagrams. These items do not have descriptions in the table associated with the diagram, but are shown in a previous diagram and described in a previous table.

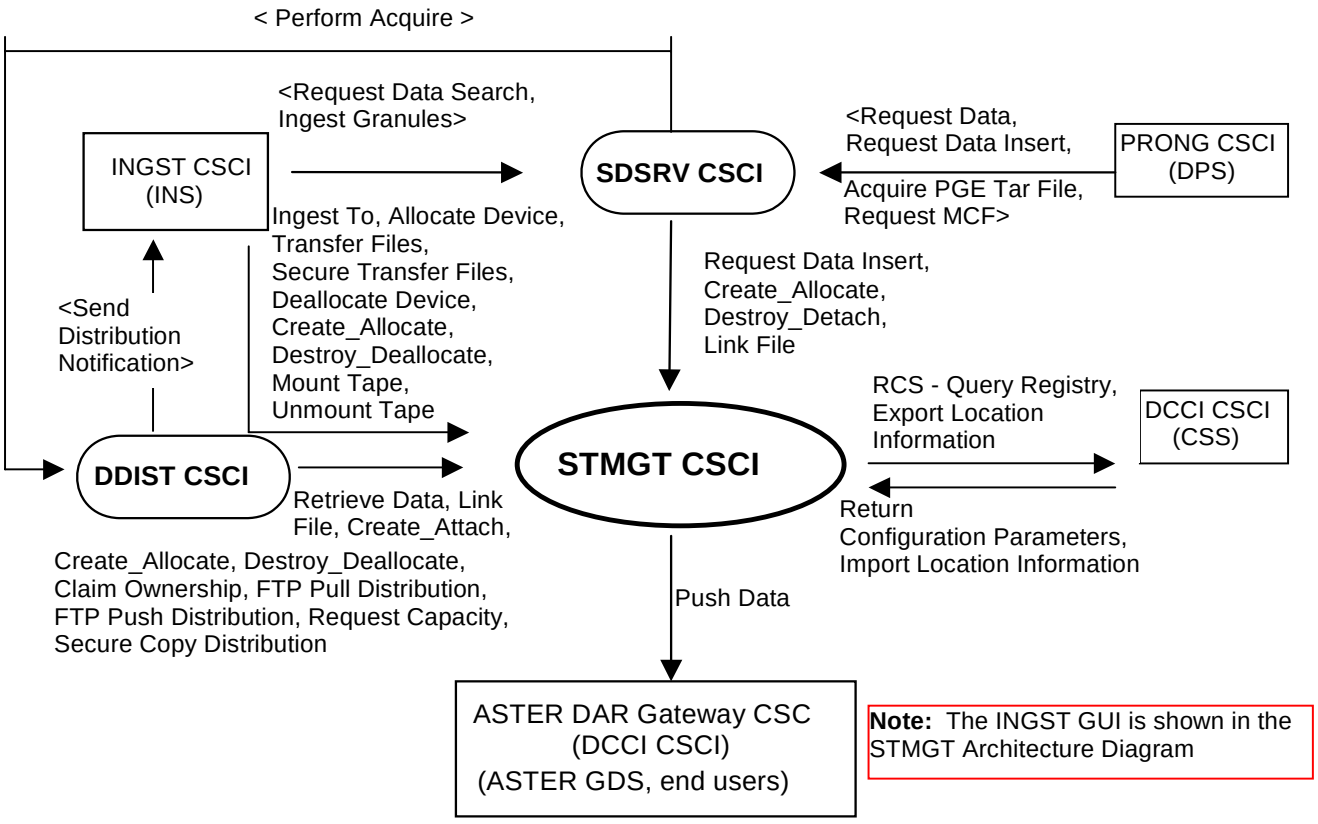


Figure 4.1.3-1. STMGT CSCI Context Diagram

Table 4.1.3-1 provides descriptions of the interface events shown in the STMGT CSCI context diagram.

Table 4.1.3-1. STMGT CSCI Interface Events (1 of 3)

Event	Interface Event Description
Request Management Services	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document.
Request Data	The **PRONG CSCI** sends data retrieval requests to the SDSRV CSCI. The SDSRV CSCI sends the data retrieval request to the DDIST CSCI. The DDIST CSCI puts the request in a queue with the appropriate priority. When the request is taken from the queue, the STMGT CSCI is passed the data retrieval request from the DDIST CSCI for a particular data granule to be pushed onto the DPS science processor, via the FTP service. The data granule is to be used as input for PGE processing or for SSIT work.
Request Data Insert	The **PRONG CSCI** sends data insert requests to the SDSRV CSCI. The SDSRV CSCI sends the data insert request to the STMGT CSCI. The STMGT CSCI receives the data insert request for a particular file or files to be inserted into the archive, and their metadata be catalogued into the SDSRV inventory, as a granule of a particular ESDT short name and version. These files can be processing output, static files received with PGEs, PGE Tape Archive (TAR) files, APs, SSAPs or DAPs, failed PGE tar files, or production history files.
Acquire PGE Tar File	The **PRONG CSCI** sends requests to the SDSRV CSCI to acquire a tar file for any PGE not currently local to the science processor. The executable is extracted from the tar file and used during PGE execution in the PRONG CSCI.
Request MCF	The **PRONG CSCI** requests the MCF from the SDSRV CSCI for a particular ESDT short name prior to a data insert request.
Create_Allocate	The STMGT CSCI receives requests from the **DDIST CSCI** and **INGST CSCI** to allocate areas on the local staging disk to store data for distribution. The SDSRV CSCI sends requests to the STMGT CSCI to allocate areas on the local staging disk to store ingested data or output files from routine data processing or SSIT work.
Destroy_Detach	The **SDSRV CSCI** sends requests to the STMGT CSCI to detach from a staging disk area (lose access to an existing staging disk area owned by another process).
Link File	The **SDSRV** and **DDIST CSCIs** sends requests to the STMGT CSCI to link files from read-only cache or a staging disk to a staging disk specified in the request.

Table 4.1.3-1. STMGT CSCI Interface Events (2 of 3)

Event	Interface Event Description
Request Communications Support	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry • Request Distribution Media Options from the Configuration Registry
Export Location Information	The CCS Middleware CSC stores physical and logical location information received from the STMGT CSCI in the **DCCI CSCI** (CCS NameServer via the Server Locator).
Return Configuration Parameters	The **DCCI CSCI** returns the requested configuration parameters to the STMGT CSCI.
Import Location Information	The STMGT CSCI requests server location information from the **DCCI CSCI (**CCS NameServer via the Server Locator).
Push Data	The STMGT CSCI pushes data (i.e., EDS), via the FTP service to the **ASTER DAR Gateway** for data distribution per user request. A signal file is also sent to indicate the completion of the file transfer by particular ESDTs that function this way.
Retrieve Data	The **SDSRV CSCI** sends requests to the STMGT CSCI to retrieve data or products from the archival storage.
Create_Attach	The STMGT CSCI receives requests from the **DDIST CSCI** to attach (gain access to an existing staging disk area allocated by another process) to a staging disk area.
Create_Allocate	The STMGT CSCI receives requests from the **DDIST CSCI** and INGST to allocate areas on the local staging disk to store data for distribution.
Destroy_Deallocate	The STMGT CSCI receives requests from the **DDIST CSCI** and **INGST CSCI** to deallocate (lose access to) an existing staging disk area.
Claim Ownership	The STMGT CSCI receives requests from the **DDIST CSCI** to claim ownership of (take responsibility for deallocating) an existing staging disk area.
FTP Pull Distribution	The STMGT CSCI receives requests from the **DDIST CSCI** to move a file to the Pull area.
FTP Push Distribution	The STMGT CSCI receives requests from the **DDIST CSCI** to distribute a file directly to the user via a predetermined local disk area accessible by the user.

Table 4.1.3-1. STMGT CSCI Interface Events (3 of 3)

Event	Interface Event Description
Request Capacity	The **DDIST CSCI** sends requests to the STMGT CSCI to get the effective maximum capacity of a media type. This information is used to determine the number of media needed to satisfy a given request.
Secure Distribution	The STMGT CSCI receives requests from the DDIST CSCI to distribute a file directly to the user securely via a predetermined local disk area accessible by the user.
Send Distribution Notification	The DDIST CSCI sends a distribution notification, via e-mail, to the INGST CSCI when data being distributed is to be ingested.
Perform Acquire	A request is sent from the **SDSRV CSCI** to the DDIST CSCI for science data or a product to be sent to a specified user. The **SDSRV CSCI** assembles instructions to send data and sends the instructions to the DDIST CSCI. The DDIST CSCI stores the request received from the **SDSRV CSCI** in a queue with the appropriate priority. The DDIST CSCI sends a request to the STMGT CSCI to push the data, via the FTP service. When the request is taken from the queue, the STMGT CSCI is passed the data retrieval request from the DDIST CSCI for a particular data granule to be pushed onto the DPS science processor, via the FTP service. The data granule is to be used as input for PGE processing or for SSIT work.
Request Data Search	The INGST CSCI sends a search request to the SDSRV CSCI for a granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Ingest Granules	The INGST CSCI sends requests to the SDSRV CSCI to insert a particular file or files into the SDSRV inventory and archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version.

4.1.3.3 Storage Management Architecture

Note: All interface events from a client to the STMGT CSCI pass through the Request Manager process. For external processes sending interface events to the Request Manager, the sequence is external process to the Request Manager to the Sybase ASE check-pointing tables to the STMGT process. For internal processes within the STMGT CSCI, the sequence is STMGT process to the Sybase ASE tables to the receiving STMGT process. Figure 4.1.3-2 below illustrates the flow of requests in an FTP Pull distribution from the DDIST CSCI to the STMGT CSCI.

The STMGT CSCI architecture diagram consists of three diagrams to better display the functionality of the STMGT CSCI. Figure 4.1.3-2 is the STMGT CSCI INGEST architecture diagram. Figure 4.1.3-3 is the STMGT CSCI DISTRIBUTION STAGING architecture diagram. Figure 4.1.3-4 is the STMGT CSCI DISTRIBUTION TRANSFER architecture diagram. The diagrams show the events sent to the STMGT CSCI processes and the events the STMGT CSCI processes sends to other processes or gateways for remote systems such as the ASTER GDS. **Note:** any items italicized and inside of < > are items which show consistency across the diagrams. These items do not have descriptions in the table associated with the diagram, but are shown in a previous diagram and described in a previous table. Figure 4.1.3-4 is the STMGT

CSCI Registry architecture diagrams showing the interaction between the STMGT CSCI and the DCCI CSCI Registry Server.

Table 4.1.3-2 provides descriptions of the interface events shown in the STMGT CSCI INGEST architecture diagram. Table 4.1.3-3 provides descriptions of the interface events shown in the STMGT CSCI DISTRIBUTION STAGING architecture diagram. Table 4.1.3-4 provides descriptions of the interface events shown in the STMGT CSCI DISTRIBUTION TRANSFER architecture diagram. Table 4.1.3-5 provides descriptions of the interface events shown in the STMGT CSCI and DCCI CSCI Registry Server architecture diagrams.

The STGMT CSCI is composed of the following processes:

- EcDsStArchiveServer (used for archiving data)
- EcDsStCacheManagerServer and EcDsStStagingDiskServer (used for staging data)
- EcDsStPullMonitorServer (used for electronic data transfer)
- EcDsStRequestManagerServer (used to route requests from clients to servers)
- EcDsStFtpServer (used to excute FtpPush ,FtpPull, route scp requests to EcDsStCopyServer)
- EcDsStCopyServer (used to execute scp requests from EcDsStFtpServer)

The STMGT GUI software (EcDsStmgtGui) is an interface with the database to set parameter configurations and to monitor and manage the servers.

The Archival Management and Storage System (AMASS) is an automated library management system. See the AMASS Release Notes for version 5.3.1 from vendor ADIC. The release notes are provided as part of Pre-Ship Review (PSR) document #914-TDA-234 for AMASS 5.3.1 released on 03/24/2003 for more information about the AMASS management and storage system.

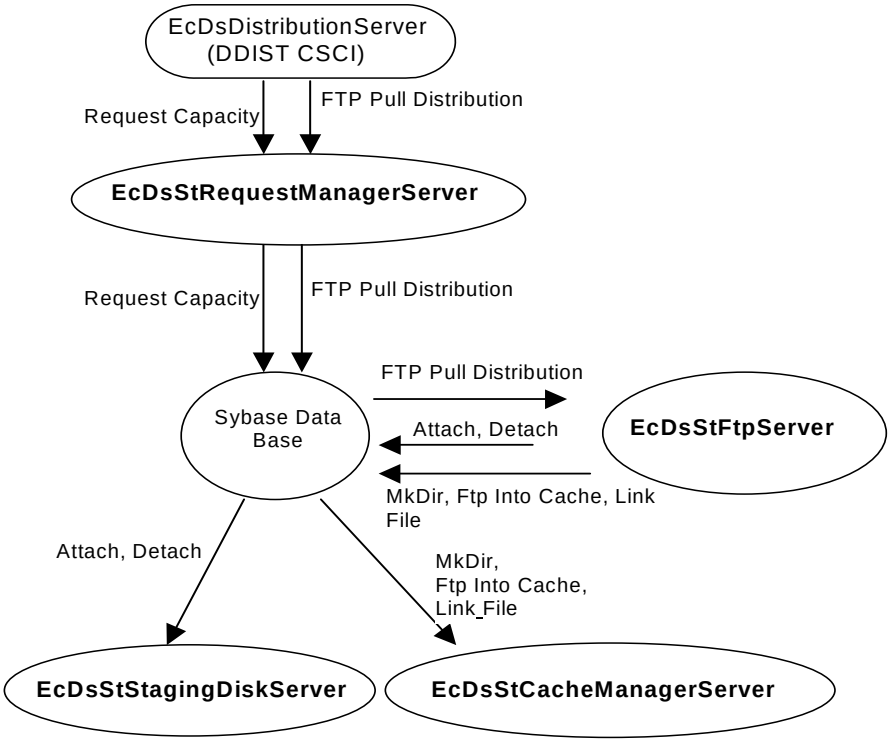


Figure 4.1.3-2. Sample FTP Pull Distribution Request

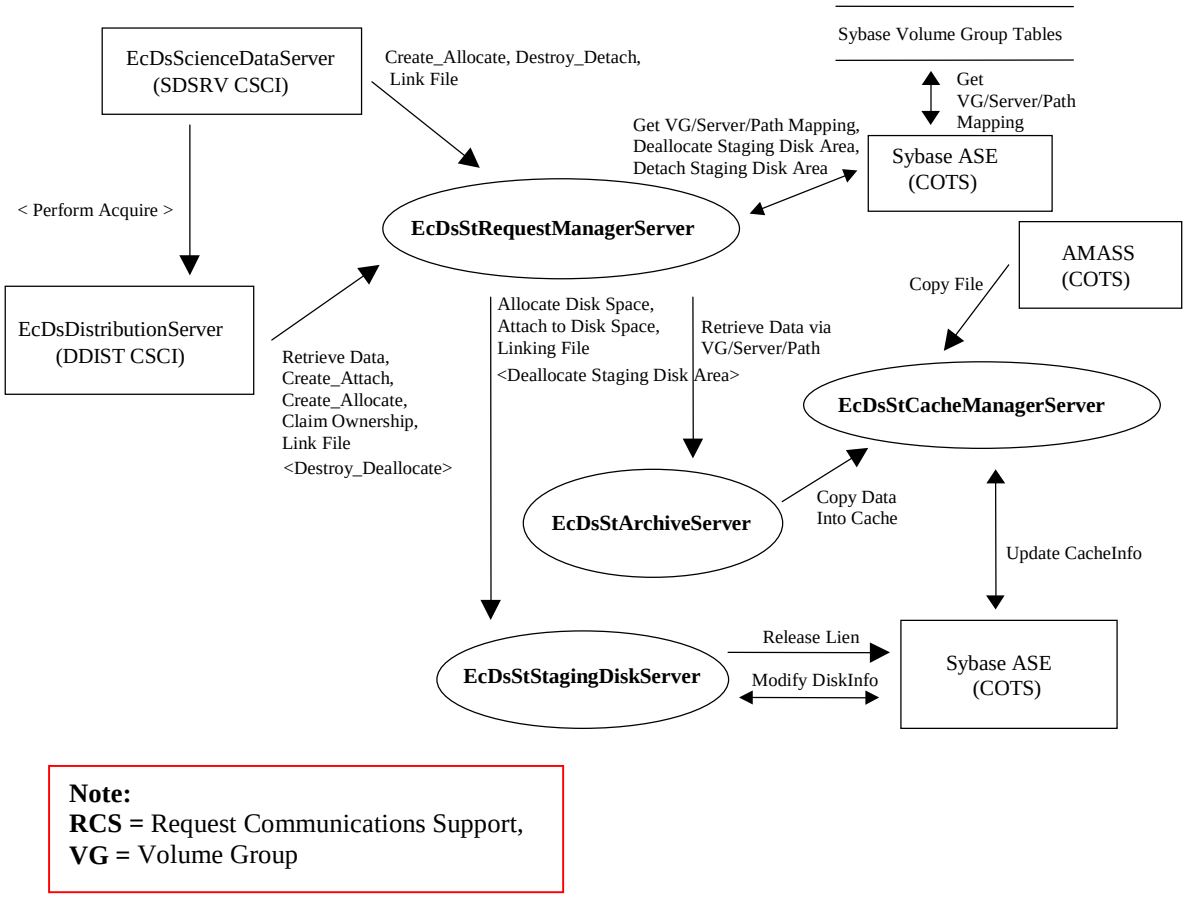


Figure 4.1.3-3. STMGT CSCI DISTRIBUTION STAGING Architecture Diagram

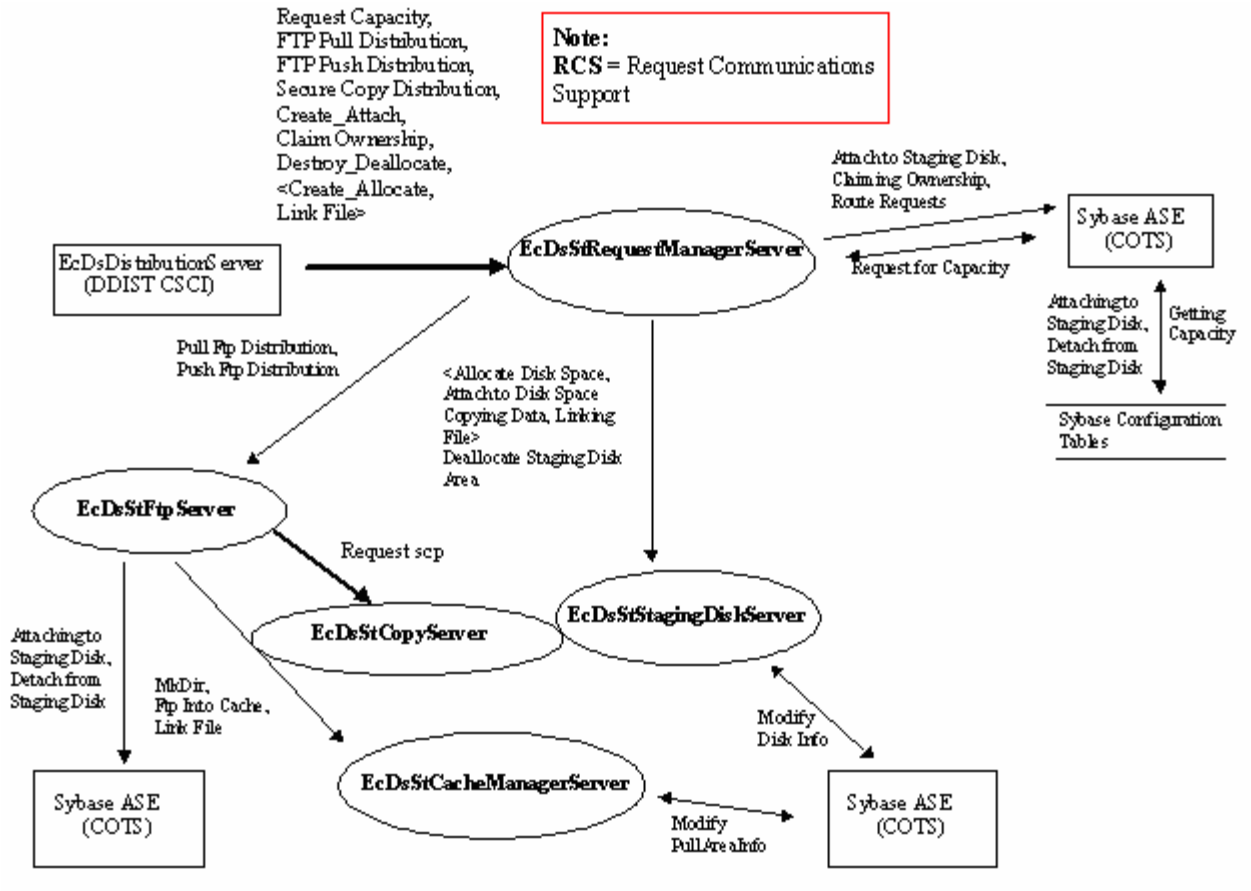


Figure 4.1.3-4. STMGT CSCI DISTRIBUTION TRANSFER Architecture Diagram

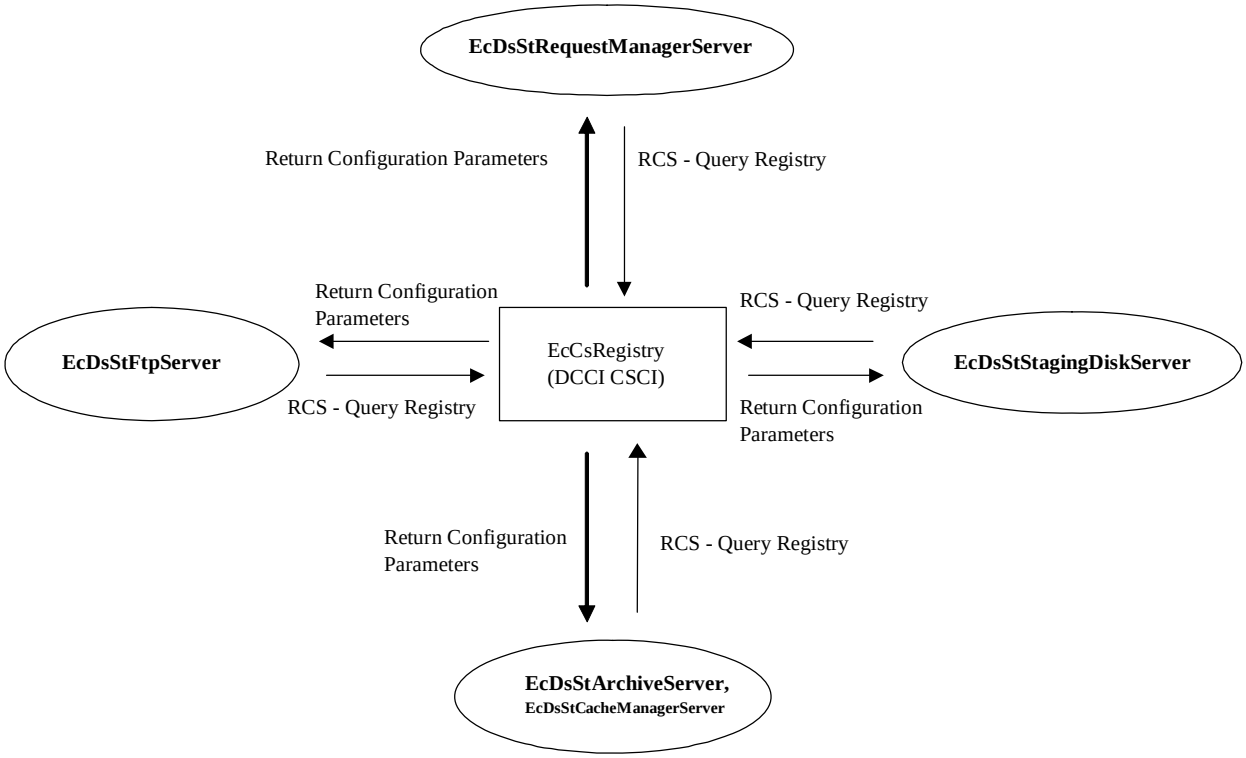


Figure 4.1.3-5. STMGT CSCI Registry Architecture Diagram

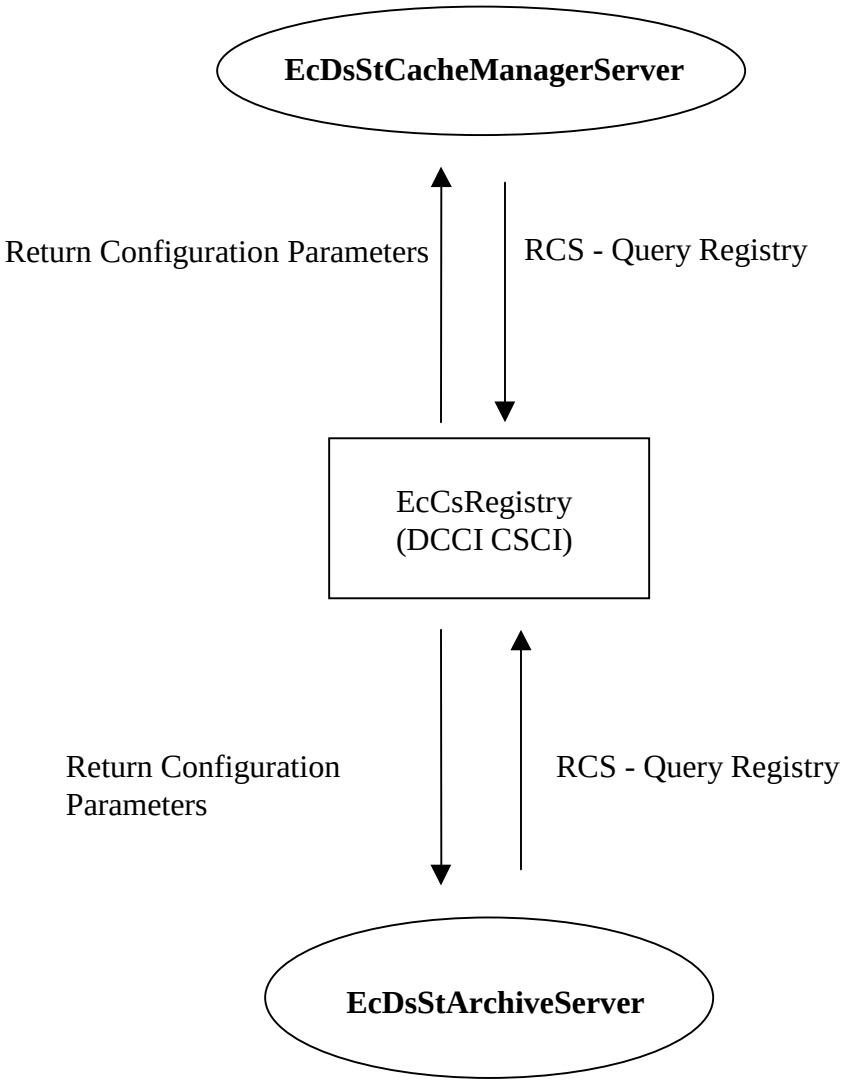


Figure 4.1.3-5. STMGT CSCI Registry Architecture Diagram (cont.)

4.1.3.4 Storage Management Process Descriptions

Table 4.1.3-2 provides descriptions of the processes shown in the STMGT CSCI INGEST, DISTRIBUTION STAGING, and DISTRIBUTION TRANSFER architecture diagrams, respectively.

Table 4.1.3-2. STMGT CSCI Processes (1 of 2)

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcDsStArchiveServer	Server	DRPHW	Developed	An Archive Server provides access to stored data. There can be multiple archive servers running at a given site, each with its own type of data or storage media. For requests retrieving files from the Archive, the staging disk is located on the same host as the Archive. For data being inserted, based on network architecture, files are moved to a staging disk located on a node with the appropriate network access. Within STMGT, ESDTs are allocated to Archive instances.
EcDsStCacheManagerServer	Server	ACMHW DRPHW	Developed	The Cache Manager Server manages the group of data files retrieved from the archive and placed into a cache area on a staging disk. A list of these data files is maintained so subsequent data retrieval requests are fulfilled immediately without requiring an additional archive access. The Cache Manager Server also deletes files, which are no longer used to prevent the cache area from becoming too full. The STMGT CSCI supports multiple cache managers.
EcDsStStagingDiskServer	Server	ACMHW DRPHW ICLHW	Developed	The Staging Disk Server manages shared disk space. The Staging Disk Server enables disk space allocations and file reservations between staging directories and from non-staging to staging directories. The STMGT software supports multiple instances of the Staging Disk Server.
EcDsStPullMonitorServer	Server	ACMHW	Developed	The Pull Monitor Server manages the files in the user pull area. As EMD users retrieve files (i.e., files are electronically pulled) from the user pull area or as the files become stale (their time-out periods have expired), the Pull Monitor Server deletes them. Several algorithms are available for monitoring and maintaining the data levels at a specified capacity. **Note:** This is just a symbolic link to the Cache Manager Server binary executable image.
EcDsStFtpServer	Server	ACMHW DRPHW ICLHW	Developed	The FTP Server manages the electronic transfer of files to and from staging disks. The STMGT software supports multiple instances of the FTP Server.
EcDsStRequestManagerServer	Server	ACMHW	Developed	The Request Manager routes requests to the appropriate server for servicing. Priority queuing is enforced for all requests, regardless of type and the status of all requests is centrally maintained in the database. The Request Manager provides the primary point of detection and recovery for unexpected client or server termination.
EcDsStCopyServer	Server	ACMHW DRPHW ICLHW	Developed	The Copy Server performs secure file copies between remote and local on behalf of the FTP Server.

Table 4.1.3-2. STMGT CSCI Processes (2 of 2)

Process	Type	Hardware CI	COTS/ Developed	Functionality
AMASS	Server	DRPHW	COTS	Provides a Unix File System interface to the robotics to control the media where data is written and read.
Sybase ASE	Server	ACMHW	COTS	The Sybase ASE interacts with other STMGT CSCI servers to copy files to be stored in the EMD archives via AMASS, to store schedules for media distribution processing, to create and delete staging disk storage areas, and to allow Operations staff to insert or update data distribution information.

In the EBIS Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.1.3.5 Storage Management Process Interface Descriptions

Table 4.1.3-3 provides descriptions of the interface events shown in the STMGT CSCI DISTRIBUTION STAGING architecture diagram.

Table 4.1.3-3. STMGT CSCI DISTRIBUTION STAGING Process Interface Events (1 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Get VG/Server/Path Mapping	One per volume group	Sybase ASE (COTS) [Database Tables: DsStVolumeGroup]	*Process:* EcDsStRequestManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStRequestManagerServer obtains the physical Unix path used to store data for the specified ESDT from the **Sybase ASE**.
Deallocate Staging Disk Space	One per attach request	Sybase ASE (COTS)	*Process:* EcDsStRequestManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStRequestManagerServer sends a request to the **Sybase ASE** to deallocate an existing staging disk area.

Table 4.1.3-3. STMGT CSCI DISTRIBUTION STAGING Process Interface Events (2 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Detach Staging Disk Area	One detach per request	*Process:* EcDsStStagingDiskServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManagerServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests to the **EcDsStStagingDiskServer** for processing.
Copy file	One file copy per request	AMASS (COTS)	*Process:* EcDsStCacheManagerServer *Library:* DsStCmnFileIO *Classes:* DsStCopyService	The EcDsStCacheManagerServer sends requests for a Unix file copy from the AMASS cache to the read-only cache by buffered read/write software.
Retrieve Data via VG/Server/Path	One per request	*Process:* EcDsStArchiveServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManagerServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests to retrieve archived data from the archive to the **EcDsStArchiveServer**.
Copy Data Into Cache	One for each file per request	*Process:* EcDsStCacheManagerServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStArchiveServer *Library:* DsStTmClient *Class:* DsStCacheManagerAsyncIF	The EcDsStArchiveServer sends requests to the EcDsStCacheManagerServer to move data from the archives to the read-only cache.
Update CacheInfo	One file copy per request	Sybase ASE (COTS) [Database Tables: DsStCache, DsStCacheFile, DsStDeleteLogCacheFile]	*Process:* EcDsStCacheManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStCacheManagerServer submits requests to the Sybase ASE to update the availability of space in the read-only cache upon receiving requests to move data from the EcDsStArchiveServer.

Table 4.1.3-3. STMGT CSCI DISTRIBUTION STAGING Process Interface Events (3 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Modify DiskInfo	Two per request (one each for allocation and deallocation)	Sybase ASE (COTS) [Database Table: DsStStagingDisk, DsStStagingDiskLien, DsStStagingDiskFile]	*Process:* EcDsStStagingDiskServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStStagingDiskServer submits a request to the Sybase ASE to update the availability of space on the local disk. The Sybase ASE updates its staging area tables (data stores).
Release Lien	One per file copied	Sybase ASE (COTS) [Database Table: DsStStagingDisk, DsStStagingDiskLien, DsStStagingDiskFile]	*Process:* EcDsStStagingDiskServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStStagingDiskServer sends requests to the Sybase ASE to decrease the access count to a given file. This is done after a file has been copied or a link to a file in the cache has been removed.
Retrieve Data	One granule per retrieval request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to retrieve data from the SDPS archives to be staged for distribution. The EcDsStRequestManagerServer forwards the requests to the **EcDsStArchiveServer**.
Create_Attach	One attach per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdMedia	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to attach (gain access) to an existing staging disk area allocated by another process.
Create_Allocate	One allocation per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	***Process:*** EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT ***Process:*** EcDsDistributionServer *Class:* DsDdMedia	The **EcDsScienceDataServer** and **EcDsDistributionServer** send requests to the EcDsStRequestManagerServer to allocate areas on the local staging disk to store data for distribution.

Table 4.1.3-3. STMGT CSCI DISTRIBUTION STAGING Process Interface Events (4 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Claim Ownership	One per attach request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdMedia	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to claim ownership of (take responsibility for deallocating) an existing staging disk area.
Copy Data	One file copy per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	***Process:*** EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT ***Process:*** EcDsDistributionServer *Class:* DsDdGranuleS	The **EcDsScienceDataServer** and **EcDsDistributionServer** send requests to the EcDsStRequestManagerServer to copy data within staging disks and between staging disks.
Link File	One link per file in a request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	***Process:*** EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT ***Process:*** EcDsDistributionServer *Class:* DsDdGranule	The **EcDsScienceDataServer** and **EcDsDistributionServer** send requests to the EcDsStRequestManagerServer to link files from the read-only cache to a staging disk and from one staging disk to another.
Allocate Disk Space	One allocation per request	*Process:* EcDsStStagingDiskServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManagerServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests to the **EcDsStStagingDiskServer**.

Table 4.1.3-3. STMGT CSCI DISTRIBUTION STAGING Process Interface Events (5 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Attach to Disk Space	One attach per request	*Process:* EcDsStStagingDiskServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManagerServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests to the **EcDsStStagingDiskServer** for processing.
Linking File	One link per file in a request	*Process:* EcDsStStagingDiskServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManagerServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests to the **EcDsStStagingDiskServer** for processing.
Destroy_Detach	One detach per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsScienceDataServer *Library:* DsGe *Class:* DsGeESDT	The **EcDsScienceDataServer** sends requests to the EcDsStRequestManagerServer to detach (lose access) to an existing staging disk area owned by another process.

Table 4.1.3-4 provides descriptions of the interface events shown in the STMGT CSCI DISTRIBUTION TRANSFER architecture diagram.

Table 4.1.3-4. STMGT CSCI DISTRIBUTION TRANSFER Process Interface Events (1 of 6)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Capacity	Per user request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to get the effective maximum capacity of a media type. This information is used to determine the number of media needed to satisfy a given request.
FTP Pull Distribution	One file per pull request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdFtpPullMedia	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to move a file to the Pull area.
FTP Push Distribution	One per push request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdFtpPushMedia	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to distribute a file directly to a user.
Secure Distribution	One per secure distribution request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdscpMedia	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to distribute a file directly to a user securely.

Table 4.1.3-4. STMGT CSCI DISTRIBUTION TRANSFER Process Interface Events (2 of 6)

Event	Event Frequency	Interface	Initiated By	Event Description
Create_Attach	One attach per request	***Process:*** EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager Sybase ASE (COTS)	*Process:* EcDsDistributionServer *Classes:* DsDdMedia, DsDsStaging	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to attach (i.e., gain access) to an existing staging disk area allocated by another process. The EcDsStRequestManagerServer sends requests to the **Sybase ASE** to attach to staging disk areas.
Claim Ownership	One per attach request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcDsDistributionServer *Class:* DsDdMedia	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to claim ownership of (i.e., take responsibility for deallocating) an existing staging disk area.
Destroy_Deallocate	One deallocation per request	***Process:*** EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager ***Process:*** EcDsStStagingDiskServer *Library:* DsStTmServer *Class:* DsStBaseReal	*Process:* EcDsDistributionServer *Class:* DsDdMedia	The **EcDsDistributionServer** sends requests to the EcDsStRequestManagerServer to deallocate (i.e., lose access to) an existing staging disk area. The EcDsStRequestManagerServer sends requests to the **EcDsStStagingDiskServer** to deallocate (i.e., lose access to) an existing staging disk area.
Attach to Staging Disk	One per request	Sybase ASE (COTS)	***Process:*** EcDsStRequestManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStRequestManagerServer sends requests to the **Sybase ASE** to attach (i.e., gain access) to an existing staging disk area.

Table 4.1.3-4. STMGT CSCI DISTRIBUTION TRANSFER Process Interface Events (3 of 6)

Event	Event Frequency	Interface	Initiated By	Event Description
Claiming Ownership	One per attach request	Sybase ASE (COTS)	*Process:* EcDsStRequestManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStRequestManagerServer sends requests to the **Sybase ASE** to claim ownership of (i.e., take responsibility for deallocating) an existing staging disk area.
Route Requests	Per external client requests	Sybase ASE (COTS) Database Tables: DsStGenericRequest, DsStArchiveRequest, DsStFtpRequest, DsStMediaRequest, DsStStagingDiskRequest	*Process:* EcDsStRequestManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	All requests submitted by external clients are executed within the **Sybase ASE** database. Requests that can be fulfilled solely by the database (“trivial requests) are satisfied and the results returned to the client with no additional communications with STMGT servers.
Request for Capacity	One calculation per request	Sybase ASE (COTS)	*Process:* EcDsStRequestManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The EcDsStRequestManagerServer sends requests to the **Sybase ASE** to determine the effective maximum capacity.
Attaching to Staging Disk	One per attach request	Sybase ASE (COTS)	*Process:* EcDsStFtpServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The **EcDsStFtpServer** send requests to the Sybase ASE to attach (i.e., gain access) to an existing staging disk area.

Table 4.1.3-4. STMGT CSCI DISTRIBUTION TRANSFER Process Interface Events (4 of 6)

Event	Event Frequency	Interface	Initiated By	Event Description
Detach From Staging Disk	One per request	Sybase ASE (COTS)	*Process:* EcDsStFtpServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The **EcDsStFtpServer** send requests to the Sybase ASE to detach (i.e., lose access to) from an existing staging disk area.
Getting Capacity	One calculation per request	Sybase ASE (COTS) Database Tables:	Sybase ASE (COTS)	The Sybase ASE obtains the capacity for a particular media type from the Sybase configuration tables.
Modify DiskInfo	One per distribution request	Sybase ASE (COTS) [Database Table: DsStStagingDisk, DsStStagingDiskLien, DsStStagingDiskFile]	*Process:* EcDsStStagingDiskServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The **EcDsStStagingDiskServer** submits a request to the Sybase ASE to update the availability of space on the local disk. The Sybase ASE updates its staging area tables (data stores).
Modify PullAreaInfo	One per file to insert	Sybase ASE (COTS) [Database Tables: DsStCache, DsStManagedCacheDir, DsStCacheFile, DsStDeleteLogCacheFile, DsStFileLink]	*Process:* EcDsStCacheManagerServer *Library:* DsStCmnDb *Class:* DsStDBIFTransaction	The **EcDsStCacheManagerServer** submits requests to the Sybase ASE to update the availability of space in the Pull cache upon receiving requests to create new user directories and transfer files (via the Ftp Service) from the EcDsStFtpServer.
MkDir	One per directory creation	*Process:* EcDsStPullMonitorServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStFtpServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStFtpServer sends requests to the EcDsStPullMonitorServer to create directories in the user pull area to which files are linked and from which the external requester can pull files.

Table 4.1.3-4. STMGT CSCI DISTRIBUTION TRANSFER Process Interface Events (5 of 6)

Event	Event Frequency	Interface	Initiated By	Event Description
Ftp Into Cache	One file per insert request into the Pull cache	*Process:* EcDsStPullMonitorServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStFtpServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStFtpServer sends requests to the EcDsStPullMonitorServer to transfer files (via the FTP service) into the Pull cache. In response, the EcDsStPullMonitorServer returns status to indicate whether the file needs to be transferred to the cache or is already resident.
Link File	One link per file in a request	*Process:* EcDsStPullMonitorServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStFtpServer *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStFtpServer sends requests to the EcDsStPullMonitorServer to link files from the Pull cache to the user pull area.
Pull Ftp Distribution	One file per pull request	*Process:* EcDsStFtpServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManager Server *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests to the **EcDsStFtpServer** for processing.
Push Ftp Distribution	One per push request	*Process:* EcDsStFtpServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManager Server *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests to the **EcDsStFtpServer** for processing.
Deallocate Staging Disk Area	One per attach request	*Process:* EcDsStStagingDiskServer *Library:* DsStTmServer *Class:* DsStReceptionist	*Process:* EcDsStRequestManager Server *Library:* DsStTmClient *Class:* DsStPatron	The EcDsStRequestManagerServer forwards the requests (deallocate staging disk area) to the **EcDsStStagingDiskServer** for processing.

Table 4.1.3-4. STMGT CSCI DISTRIBUTION TRANSFER Process Interface Events (6 of 6)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Management Services	At system startup or shutdown and for restarts	*Processes:* EcDsStRequestManagerServer, EcDsStArchiveServer, EcDsStStagingDiskServer, EcDsStCacheManagerServer, EcDsStFtpServer,	DAAC unique startup scripts	**System startup and shutdown –** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document.

Table 4.1.3-5 provides descriptions of the interface events shown in the STMGT CSCI Registry architecture diagram.

Table 4.1.3-5. STMGT CSCI Registry Process Interface Events (1 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Communications Support	One service per request	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Library:*** event *Class:* EcLgErrorMsg ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Processes:* EcDsStRequestManagerServer, EcDsStCacheManagerServer, EcDsStStagingDiskServer, EcDsStFtpServer	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry – Retrieving the requested configuration attribute-value pairs from the Configuration Registry

Table 4.1.3-5. STMGT CSCI Registry Process Interface Events (2 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Configuration Parameters	One set per request	*Processes:* EcDsStRequestManagerServer, EcDsStArchiveServer, EcDsStCacheManagerServer, EcDsStStagingDiskServer, EcDsStFtpServer *Library:* EcCsRegistry Class: EcRgRegistryServer_C	*Process:* EcCsRegistry *Library:* EcCsRegistry Class: EcRgRegistryServer_C	The EcCsRegistry returns the attribute-value pairs (configuration parameters) to the **STMGT processes – EcDsStRequestManagerServer, EcDsStArchiveServer, EcDsStCacheManagerServer, EcDsStStagingDiskServer,** and **EcDsStFtpServer** – upon request.

4.1.3.6 Storage Management Data Stores

Table 4.1.3-6 provides descriptions of the individual data stores (entitled collectively "Database") in the STMGT CSCI architecture diagram. More detailed information on these database tables can be found in the CDRL 311 document on Storage Management (CD-311-625).

Table 4.1.3-6. STMGT CSCI Data Stores (1 of 4)

Data Store	Type	Description
Sybase Cache Tables		
DsStCache	Sybase	This data store identifies every instance of a read-only cache or Pull Area. The current design supports multiple read-only caches, but only one Pull Area.
DsStCacheFile	Sybase	This data store contains an entry for each file in a STMGT cache (read-only cache or pull area). An entry is inserted into the entity for each file retrieved from the Archival Management And Storage System (AMASS).
DsStDeleteLogCacheFile	Sybase	This data store contains a historic record of each file the STMGT CSCI deletes from its caches (or the DsStCacheFile table). This entity maintains a history of file usage and cache usage for reporting and analysis purposes. An entry is inserted into the entity via a delete trigger in the DsStCacheFile table.
DsStNotification	Sybase	This data store contains configuration information for Pull Area monitoring purposes.
DsStFileLien	Sybase	This data store contains the names of files that have been staged in read-only cache in preparation for being copied or linked to a staging disk. Liens are removed upon copy and converted to links upon link. Liens expire and are automatically removed if not exercised within a configured number of hours.

Table 4.1.3-6. STMGT CSCI Data Stores (2 of 4)

Data Store	Type	Description
DsStFileLink	Sybase	This data store contains the file links to user Pull Area directory or the read only cache.
DsStManagedCacheDir	Sybase	This data store contains an entry for each user directory in the Pull Area. An entry is inserted into the data store every time a new directory is added to the Pull Area.
DsStPending Reservations	Sybase	This data store contains a queue for attempts to reserve space, which cannot be met at the time. The queue is serviced on a First-In/First Out (FIFO) priority order.
DsStPendingDelete	Sybase	This data store is used to record granule files ready to be deleted. Storage Management reads this table and deletes the corresponding information from the database.
Sybase Staging Area Table		
DsStStagingDisk	Sybase	This data store saves information (e.g., staging disk size, owner, disk number, and priority request id) about requests for staging disk services.
DsStStagingDiskLien	Sybase	This data store contains a list of the clients currently attached to staging disks. Staging disks are not removed until all liens are released or expired.
DsStStagingDiskFile	Sybase	This data store contains the file names copied to the staging disk areas. This data store is used to track links between staging disks.
Sybase Volume Group Tables		
DsStBackup	Sybase	Stores a reference to every file currently being backed-up related to Archive Backup and Restore functionality.
DsStBackupHistory	Sybase	Stores a history of every file successfully backed-up related to Archive Backup and Restore functionality.
DsStVolumeGroup	Sybase	This data store contains 'volume group' information (the section of the EMD Archive being accessed) from configuration files such as the path currently pointed to and a history of paths used to store data of a given ESDT and version.
Sybase Configuration Tables (All Servers Use these Tables)		
DsStConfigParameter	Sybase	This data store contains an entry for information to configure and initialize each server instance supported by Storage Management. The data consists of information currently accessed through configuration files (*.CFG) plus information as it pertains to the status and node of operation for each server. An entry is inserted for each parameter that a server uses. Two types of parameters can be defined within the constructs of the DsStConfigParameter table, startup and run-time. Startup parameters require the associated server be restarted in order for the parameters to be used. Run-time parameters can be changed without restarting the server (i.e., the server periodically queries the configuration table for new values).

Table 4.1.3-6. STMGT CSCI Data Stores (3 of 4)

Data Store	Type	Description
DsStServerType	Sybase	This data store contains all types of servers administered and configured by Storage Management and their associated descriptions. Eight types of standard servers are currently pre-populated with the database construction due to no user interface currently existing to administer (e.g., Archive, Cache Manager, Staging Disk, Request Manager, Pull Monitor and FTP Server).
DsStServiceThreadConfig	Sybase	This data store contains the allocation of service threads by priority for each server. For servers that use multiple thread pools, the priority allocation of each thread pool is stored.
DsStStagingDiskServer	Sybase	This data store contains configuration parameters specific to instances of the Staging Disk Server.
DsStFtpServer	Sybase	This data store contains configuration parameters specific to instances of the FTP Server.
DsStArchiveServer	Sybase	This data store contains configuration parameters specific to instances of the Archive Server.
EcDbDatabaseVersions	Sybase	Contains version information about the installed database. Data includes the date of installation, the version number of the database installed, and the latest version number available for the loaded database.

Table 4.1.3-6. STMGT CSCI Data Stores (4 of 4)

Data Store	Type	Description
DsStPreconfiguredDevice	Sybase	This data store contains a list of devices known to the deployed version of the EMD software. This is provided as a convenience to the operator in order to facilitate installation and configuration of new hardware.
Sybase Operator Logging Tables		
DsStErrorAttribute	Sybase	This data store is required for the DsStErrorDetails class. This data store provides a mapping between character mnemonics and numeric error codes. It defines the attributes for each error, and provides adequate characterization for appropriate retry/recovery procedures from the error attributes.
DsStErrorText	Sybase	This data store provides text descriptions and suggested recovery actions for each error code, and presents errors in a meaningful manner.
DsStEventLog	Sybase	This data store contains a history of events and COTS errors encountered by the STMGT CSCI. The STMGT CSCI inserts a new ERROR_LOG entry each time an event occurs or an error is encountered. The Operations staff has the capability to purge this entity periodically based on a date/time value.
Sybase Check-pointing Tables (All Servers use these Tables)		
DsStArchiveRequest	Sybase	Stores all requests for archive services until they are completed.
DsStFile	Sybase	Contains an entry for each file STGMT is currently processing related to a DsStArchiveRequest entry.
DsStFtpRequest	Sybase	Stores information (e.g., pull directory name and parameter list for FTP request) about FTP specific requests.
DsStGenericRequest	Sybase	Stores common information to all STMGT requests (e.g., checkpoint state, request status, and type of operation, and owner name) regardless of type.
DsStCacheManagerRequest	Sybase	Contains information (e.g., unique file name, and original file name and file size for Cache Manager specific requests.
DsStCancelledRequest	Sybase	Stores a list of requests, which have been cancelled, with the reason for cancellation.
DsStDependentRequest	Sybase	Tracks dependencies between requests, i.e., where a higher-level request is waiting for a lower-level request to complete.
DsStArchiveFileRequest	Sybase	Contains information specific to Archive Server requests, which operate on a single file, e.g., ArStoreFile and ArRetrieveFile.
DsStStagingDiskRequest	Sybase	Contains information specific to Staging Disk Server requests.

4.1.4 Data Server Subsystem Hardware

4.1.4.1 Access Control and Management Hardware CI Description

The Access Control and Management HWCI (ACMHW), provides access to the Data Server subsystem for subsystem and direct “push/pull” user access, provides tools and capabilities for

system administration, and supports the infrastructure of the Data Server. This HWCI controls logical data server access, maintains sessions, provides sub-setting support, directs service requests to other appropriate Data Server Subsystem configuration items, and supports the control and data flow for electronic distributions. The Access Control and Management hardware (ACMHW) is logically divided into the Administration Stations (AS) and the Access/Process Coordinators (APCs). The number, type, and configuration of Ass and APCs depend on site requirements and the number of data servers supported. Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the ACMHW and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.1.4.2 Data Repositories Hardware CI Description

The Data Repositories (DRs) HWCI (DRPHW) is hardware to store and maintain data permanently. Different technologies are used to instantiate DRs depending on the volume and type of data to be stored, the access patterns of the data, and additional unique requirements imposed on the repository (i.e., data maintenance requirements, backup and restore functions, media management and control, etc.).

DRs are classified as “permanent”, meaning the services to monitor and maintain data integrity for large data holdings are supported by this repository's storage technology. A copy of all data at a site not considered temporary is eventually maintained in a site permanent DR.

The File and Storage Management System (FSMS) host platform is an SGI machine. The Data Server's servers and workstations are directly connected to the DAAC FDDI network and High-speed Gig Ethernet fabric.

Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the Data Repository HWCI and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.2 Ingest Subsystem Overview

The Ingest Subsystem (INS) ingests data into Science Data Processing Segment (SDPS) repositories in accordance with approved ICDs. Data is accepted from a variety of external data providers in a variety of formats predefined within SDPS regarding the expected metadata and metadata characteristics. The INS supports a variety of Ingest requests and preprocesses the data for archiving into the SDPS. The preprocessing depends on the attributes of the ingested data such as data type, data format, and the level to which the ingested data has been processed from raw instrument data.

The data types are formally referred to as Earth Science Data Types (ESDTs). An ESDT is a defined data set associated with a given mission/instrument or identified grouping and is registered in an SDPS ESDT Baseline List. The ESDT is identified with a short name, a long name, a collection description, and information on file type, metadata, formats, and services provided such as sub-setting by a given attribute. An example of an ESDT is AST_L1A; the short name for the ASTER reconstructed Level 1A data set, unprocessed instrument digital counts with radiometric (LR) and geometric (LG) coefficients attached. ESDTs can also be ancillary data, algorithms, correlative and calibration data.

The INS software processes execute in a specific pattern based on the ESDT being ingested. The processes include a polling ingest process, a media ingest process, and a cross mode ingest process. Ingest processes provide for the receipt of external data, which is archived within the EMD SDPS archival system. Specific ingest process procedures are established to support each unique INS interface and allow the processing interface parameters to be modified as interface and mission requirements evolve. For a given incoming data set corresponding to an ESDT, the INS performs data preprocessing, metadata extraction, and directs the DSS SDSRV CSCI to perform metadata validation.

Data is staged (prepared for transfer) to one of two areas depending on the data level, ESDT, and other data set specific characteristics:

- Level 0 (L0) data received from external data providers and other selected data (EDOS ancillary data) is staged to the INS working storage area. Metadata is extracted and the format is validated in the working storage area. The L0 data is transferred to an archive data repository in the DSS for long-term storage.
- Non-L0 data (such as non-EDOS ancillary data and L1A - L4 data from external facilities) is staged directly to the working storage area in the DSS. Extraction of metadata is performed on the data by the INS software residing in the INS processor hardware. The INS software residing in the INS processor hardware calls the DSS (SDSRV CSCI) to perform metadata validation. The non-L0 data is transferred to a DSS archive data repository for long-term storage.

Ingest Subsystem Context

Figure 4.2-1 is the INS context diagram. The diagram shows the events sent to the INS from other SDPS or CSMS subsystems and the events the INS sends to other SDPS or CSMS subsystems, the Operations staff, and external providers. Table 4.2-1 provides descriptions of the interface events shown in the Ingest Subsystem context diagram.

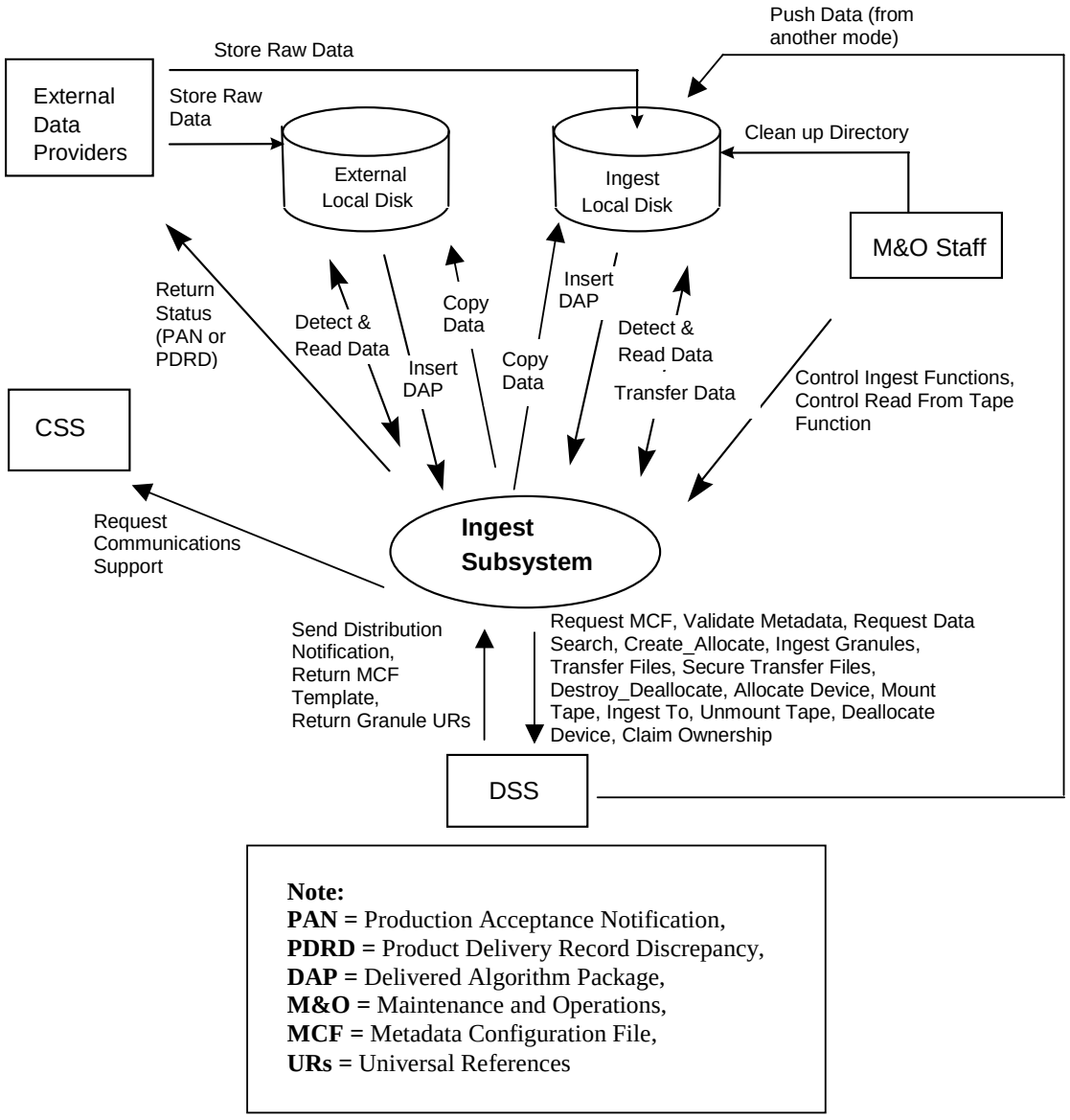


Figure 4.2-1. Ingest Subsystem Context Diagram

Table 4.2-1. Ingest Subsystem Interface Events (1 of 2)

Event	Interface Event Description
Push Data (from another mode)	The DSS pushes data, via the FTP service, to the **Ingest local disk** when it is distributing data to be ingested.
Clean up Directory	The **Operations staff** sends clean up instructions to the Ingest Local Disk for data clean up. Currently, delete and remove Unix commands are executed from the Unix command line to clean up the Ingest local disk.
Control Ingest Functions	The **Operations staff** controls the Ingest function by monitoring requests, canceling requests and granules, resuming suspended requests and granules, changing database parameters, viewing history, and performing manual media ingest via a GUI.
Control Read From Tape Function	The **M&O staff** controls the read from tape function by selecting the option on the GUI to read from tape.
Detect & Read Data	The Ingest Subsystem (INS) polls for data files, Delivery Record files, or distribution notification files in an agreed upon location (**External to EMD** or **Ingest internal Local Disk**). The Data Provider can specify using normal FTP or secure shell to transfer Product Delivery Record and data files. The Data Provider can also specify the checksum type and checksum value of data files.
Transfer Data	The INS retrieves data from the **Ingest local disk** and stores distribution notification files and PDRs for cross mode ingest on the Ingest local disk.
Insert DAP	Delivered Algorithm Packages (DAPs) are located on a **Local Disk (**external or internal to a DAAC) and are inserted into the SDPS via the automated polling ingest interface.
Copy Data	The INS copies data from tape to the **Ingest external** and/or **internal local disk**.
Request MCF	The INS requests the Metadata Configuration File (MCF) template, from the **DSS**, for a particular ESDT short name prior to a data insert request.
Validate Metadata	The **DSS** validates the metadata files that the INS has populated.
Request Data Search	The INS requests a search, by the **DSS**, for the granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Create_Allocate	The INS sends requests to the **DSS** to allocate areas on the local staging disk to store ingested data.
Ingest Granules	The INS sends requests to the **DSS** to insert a particular file or files into the SDSRV inventory and archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. The file checksum type and file checksum value will be passed along to DSS for storage and verification if specified by data provider.
Transfer Files	The INS sends requests to the **DSS** to transfer (copy) data files to a staging disk.
Secure Transfer Files	The INS sends requests to the **DSS** to secure transfer (scp) data files to a staging disk.
Destroy_Deallocate	The INS sends requests to the **DSS** to deallocate a staging disk area (to remove an existing staging disk area from usage). Deallocation of staging disks deletes transferred copies of the data after it (the original data) has been archived.
Allocate Device	The INS sends requests to the **DSS** to allocate peripheral devices for data ingesting.

Table 4.2-1. Ingest Subsystem Interface Events (2 of 2)

Event	Interface Event Description
Mount Tape	The INS sends requests to the **DSS** to load tapes to hardware peripherals for reading the tapes.
Ingest To	The INS sends requests to the **DSS** to copy files from peripheral resources to staging disk areas.
Unmount Tape	The INS sends requests to the **DSS** to unload and detach tapes from hardware peripherals after reading or writing to the tapes.
Deallocate Device	The INS sends requests to the **DSS** to deallocate the previously allocated media resource.
Claim Ownership	The INS sends requests to the **DSS** to claim ownership of (take responsibility for deallocating) an existing staging disk area.
Send Distribution Notification	The **DSS** sends a distribution notification, via e-mail, to the INS when data being distributed is to be ingested.
Return MCF Template	The **DSS** sends the MCF template to populate as part of the GetMCF service call to the INS.
Return Granule URs	The **DSS** returns the Earth Science Data Type (ESDT) Universal References (URs) for the requested granules from the INS.
Request Communications Support	The **CSS** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the CSS. These services include: • CCS Middleware Support • Database Connection Services • File Transfer Services • Network & Distributed File Services • Bulk Data Transfer Services • Name/Address Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Return Status (PAN or PDRD)	The INS returns the status of a request received from an **External Data Provider** by transmitting a Production Acceptance Notification (PAN) or a Product Delivery Record Discrepancy (PDRD) to the External Data Provider directly. The data provider can specify using normal ftp or secure copy to transfer the PAN or PDRD file or/and have it emailed to a specified email account.
Store Raw Data	The raw data (L0) provided from the **External Data Provider** to the EMD. Some external providers put this data on an external local disk for EMD to pull while others push the data onto a local INS internal disk.
Request Management Services	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document.

Ingest Subsystem Structure

The INS is one CSCI and one HWCI. Ingest backup hardware runs in the Test Mode as long as there are no failures. The configuration items are:

- Ingest (INGST) CSCI provides the software capability to acquire data by various methods and transfers the data into the SDPS. These methods include Polling with or without Delivery Record for data placed at predetermined locations, a Media transfer method, which includes reading tapes and a cross mode ingest method. The INGST CSCI also stores and manages request information, and provides for data preprocessing and insertion into the appropriate SDPS storage location.
- Ingest Client HWCI (ICLHW) supports INGST in bringing data into the SDPS from an external interface. This HWCI also includes an Ingest Workstation for execution of the Ingest GUI.

Use of COTS in the Ingest Subsystem

- Rogue Wave's Tools.h++

  The Tools.h++ class libraries provide basic functions and objects. This library and the following Rogue Wave libraries are delivered as static libraries with EMD custom code.

- Rogue Wave's DBTools.h++

  The DBTools.h++ class libraries interact with the Sybase ASE. These libraries must be installed with the INS software to interact with Sybase. The INS uses an interface software process control file (PCF) to obtain access to the Sybase ASE.

- Rogue Wave's ToolsPro.h++

  Net.h++ is included in the C++ class libraries of this tool to provide an object-oriented interface to Inter-Process Communication (IPC) and network communication services. The Net.h++ framework enables developed code to be portable to multiple operating systems and network services. These libraries must be installed with the INS software to support the interface to other subsystems.

- Integrated Computer Solutions'(ICS) Builder Xcessory (built on X/Motif)

  The Builder Xcessory GUI builder tool modifies the displays of Ingest GUIs. The tool also generates the C++ code to produce Ingest GUIs at run time. There is no operational part of the Builder Xcessory needed at run-time.

- Sybase (OpenClient, ASE)

  This set of Sybase products provides a relational database to store INS related information and must be installed on the platform where INS software resides.

- CCS Middleware Client

  This product provides the communications between INS and other subsystems. CCS Middleware can reside on one or both sides of the interface and must be installed on the platform where the INS resides. Although the CCS Middleware Client is part of the CSS, this COTS product must be installed on the platform where the INS software resides for INS to run in the EMD operational and test environments.

- UNIX Network Services

  DNS, NFS, E-mail, FTP, TCP/IP and the other Unix services provided are obtained from the CSS and are described in section 4.8 of this document.

Error Handling and processing

The INGEST Subsystem (INS) uses the class EcUtStatus for general error reporting. It is used mostly as a return value for functions and allows detailed error codes to be passed back up function stacks.

DsShError is a Science Data Server specific class used by INGEST for printing the line number in the code where an error occurred.

The INGST CSCI uses two main mechanisms for error handling.

1. Return Values

   Functions can return an EcUtStatus object, which can be used to indicate a general success/failure status, as well as more detailed information on the exact reason for the failure. This is the most widely used mechanism within INGST and in general, these errors get propagated back up to the top-level functions with ALOG error messages being generated along the way.

2. Debug Messages

   The Ingest code uses the PF_STATUS, PF_VERBOSE and PF_DEBUG macros to log messages to the debug logs. These macros are part of the EcPfGenProcess class.

For writing messages to the debug log, the following macros are used:

PF_STATUS writes a message at a "log level" of 1 to the debug log. For example,

PF_STATUS { cerr << "InDataPreprocessTask::Preprocess successful. ";

cerr << "RequestID="<< myRequestID<< " GranuleID="<<myGranuleID <<" CollectionName="<<myDataType<< endl; }

PF_VERBOSE writes a message at a "log level" of 2 to the debug log. For example,

PF_VERBOSE { cerr << "InFDDorbitMetadata::Preprocess() " << endl;

cerr << "Byte-ordered FDD Orbit Metadata preprocessing successful. ";

```
cerr << "RequestID="<< myRequestID << ", ";
cerr << "GranuleID="<< myGranuleID << ", ";
cerr << "CollectionName=" << myDataType.data() << endl;
cerr << "myInputFile: " << myInputFile.data() << endl; }
```

PF_DEBUG writes a message at a "log level" of 3 to the debug log. For example,

```
PF_DEBUG {    cerr << __FILE__ << ":" << __LINE__ << endl;
cerr << "InDataPreprocessTask::PreprocessSceneMetadata ";
cerr << "browseFileName = " << browseFileName << endl;
cerr << "DataType = " << myDataType ;
cerr << "VersionID = " << myVersionID;
cerr << "RequestID="<< myRequestID<< " GranuleID="<<myGranuleID << endl; }
```

In addition, INGST generates the following special messages: PDRD and PAN.

3. Product Delivery Record Discrepancy (PDRD)/Data Availability Acknowledgement (DAA)

   When there is a problem with a Product Delivery Record (PDR) (or Data Availability Notice (DAN), INGST generates a PDRD or DAA message. The message can be sent via e-mail and/or transferred (via FTP) if so configured. The code to generate these messages is in the InMsg library.

4. Production Acceptance Notification (PAN)/Data Delivery Notice (DDN) messages

   When an INGST request has completed, INGST generates a PAN or DDN message. The message can be sent via e-mail and/or transferred (via FTP) if so configured. The code to generate these messages is in the InMsg library.

4.2.1 INGST Computer Software Configuration Item Description

4.2.1.1 INGST Functional Overview

The INGST CSCI supports a variety of interfaces to external systems. The application-level protocol set up for data transfer is potentially different for each of the external interfaces. As a result, a separate ingest software application is required to facilitate data transfer for each interface. To minimize the software development effort and make it easier to accommodate interfaces to new external systems, data ingest from external systems is categorized, based on common characteristics and ingest processes.

4) Automated Polling Ingest Interface

- Polling with Delivery Record – The SDPS periodically checks an agreed-upon network location for a Delivery Record file. The Delivery Record describes the location of the available data. The data location could be on a working storage device within SDPS, where an external data provider has previously transferred the data.
- Polling without Delivery Record – The SDPS periodically checks an agreed-upon network location for available data. All data in the location make up a collection of ingest data of one specific ESDT, with one file per data granule.

4) Manual Media Ingest Interface (via a GUI)

Manual data transfer mechanisms – data can be transferred from physical media. Physical Media Ingest enables authorized institutions or science users to provide data on hard media. The hard media must contain information identical to the Delivery Records described above, in a standard file format, or the data provider must separately provide Delivery Records to a specified SDPS location in the standard file format. Hard media data transfer involves data transfer from one of several ingest peripheral types found at a DAAC.

3) Cross Mode Ingest Interface

The INGST CSCI receives a distribution notice, via e-mail, of data files transferred, via the FTP service. The distribution notification is used to create a Delivery Record File (describes the location of the available data). The Delivery Record file is put in an agreed-upon network location. The polling with Delivery Record process checks the location for the Delivery Record files.

The INGST CSCI includes the processes for ingesting data as described and provides a process for managing requests, and for inserting granule data into the SDPS.

4.2.1.2 INGST Context

Figures 4.2-2 is the INGST CSCI context diagrams. The diagrams show the events sent to the INGST CSCI and the events the INGST CSCI sends to other CSCIs. Table 4.2-2 provides descriptions of the interface events shown in the INGST CSCI context diagrams.

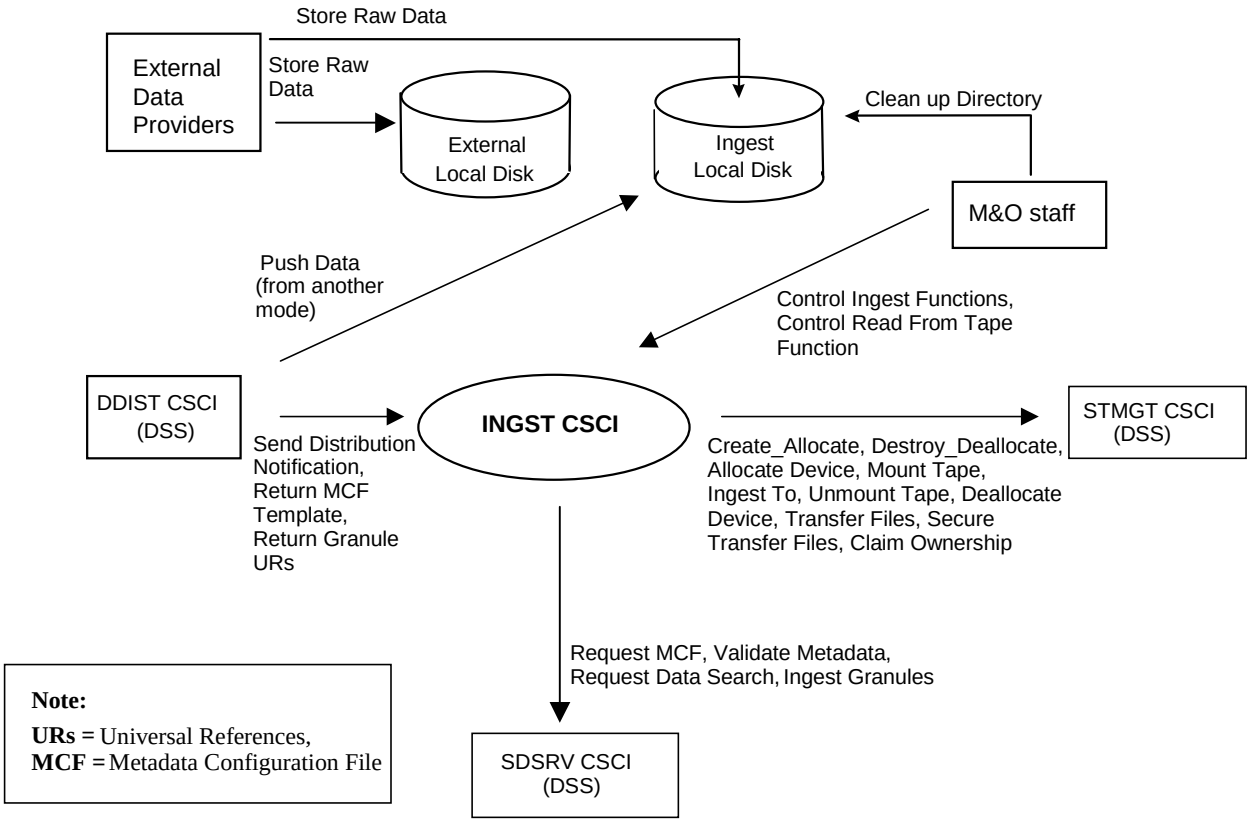
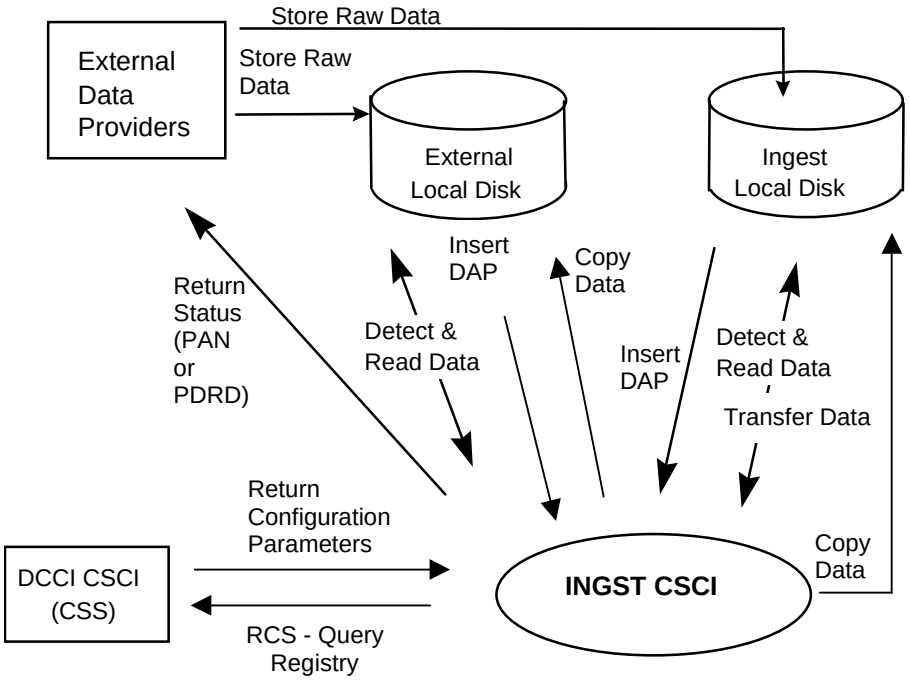


Figure 4.2-2. INGST CSCI Context Diagram



Note:
DAP = Delivered Algorithm Package,
RCS = Request Communications Support,
URs = Universal References,
MCF = Metadata Configuration File,
PAN = Production Acceptance Notification,
PDRD = Product Delivery Record Discrepancy

Figure 4.2-2. INGST CSCI Context Diagram (cont.)

Table 4.2-2. INGST CSCI Interface Events (1 of 3)

Event	Interface Event Description
Clean up Directory	The **M&O staff** sends clean up instructions to the Ingest local disk for data clean up. Currently, delete and remove Unix commands are executed from the Unix command line to clean up the Ingest local disk.
Control Ingest Functions	The **M&O staff** control the Ingest function by monitoring requests, canceling requests and granules, resuming suspended requests and granules, changing database parameters, viewing history, and performing manual media ingest via a GUI.
Control Read From Tape Function	The **M&O staff** controls the read from tape function by selecting the option on the GUI to read from tape.
Create_Allocate	The INGST CSCI sends requests to the **STMGT CSCI** to allocate areas on the local staging disk to store ingested data.
Destroy_Deallocate	The INGST CSCI sends requests to the **STMGT CSCI** to deallocate a staging disk area (to remove an existing staging disk area from usage). Deallocation of staging disks deletes transferred copies of the data after it (the original data) has been archived.
Allocate Device	The INGST CSCI sends requests to the **STMGT CSCI** to allocate peripheral devices for data ingesting.
Mount Tape	The INGST CSCI sends requests to the **STMGT CSCI** to load tapes to hardware peripherals for reading the tapes.
Ingest To	The INGST CSCI sends requests to the **STMGT CSCI** to copy files from peripheral resources to staging disk areas.
Unmount Tape	The INGST CSCI sends requests to the **STMGT CSCI** to unload and detach tapes from hardware peripherals after reading or writing to the tapes.
Deallocate Device	The INGST CSCI sends requests to the **STMGT CSCI** to deallocate the previously allocated media resource.
Transfer Files	The INGST CSCI sends requests to the **STMGT CSCI** to transfer (copy) data files into the STMGT CSCI staging disks.
Secure Transfer Files	The INGST CSCI sends requests to the **STMGT CSCI** to secure transfer (scp) data files into the STMGT CSCI staging disks.
Claim Ownership	The INGST CSCI sends requests to the **STMGT CSCI** to claim ownership of (take responsibility for deallocating) an existing staging disk area.
Request MCF	The INGST CSCI requests the Metadata Configuration File (MCF) template, from the **SDSRV CSCI**, for a particular ESDT short name prior to a data insert request.
Validate Metadata	The INGST CSCI populates the metadata files and sends them to the **SDSRV CSCI** for validation.
Request Data Search	The INGST CSCI requests a search, by the **SDSRV CSCI**, for the granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Ingest Granules	The INGST CSCI sends requests to the **SDSRV CSCI** to insert a particular file or files into the archive. Inserted data is accompanied by metadata. The metadata is catalogued in the SDSRV inventory as a granule of a particular ESDT short name and version. Inserted data could also be accompanied by file checksum information if it is specified and activated by data provider.

Table 4.2-2. INGST CSCI Interface Events (2 of 3)

Event	Interface Event Description
Send Distribution Notification	The INGST CSCI receives distribution notification, via e-mail, from the **DDIST CSCI** when data being distributed is to be ingested.
Return MCF Template	The INGST CSCI receives the MCF template to populate as part of the GetMCF service call to the **SDSRV CSCI**.
Return Granule URs	The INGST CSCI receive Earth Science Data Type (ESDT) Universal References (URs) for the granules from the **SDSRV CSCI**.
Push Data (from another mode)	The DDIST CSCI pushes data, via the FTP service, to the **Ingest local disk** when it is distributing data to be ingested.
Store Raw Data	The raw data (L0) is provided from the **External Data Provider** to the SDPS. Some External Data Providers put this data on an external local disk for SDPS to pull while others push the data onto a local INGST CSCI internal disk.
Insert DAP	Delivered Algorithm Packages (DAPs) are located on a **local disk** (**external** or **internal** to a DAAC) and are inserted into the SDPS via the automated polling Ingest interface.
Copy Data	The INGST CSCI copies data from tape to the **Ingest external** and/or **internal local disk**.
Detect & Read Data	The INGST CSCI polls for data files, Delivery Record files, or distribution notification files in an agreed upon location (**External to EMD** or **Ingest internal local disk**). The Data Provider can specify using normal FTP or secure shell to transfer the Product Delivery Record (PDR) and data files.
Transfer Data	The INGST CSCI retrieves data from the **Ingest local disk** and stores distribution notification files and PDRs for cross mode ingest on the Ingest local disk.
Request Communications Support	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • File Transfer Services • Network & Distributed File Services • Bulk Data Transfer Services • Name/Address Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Return Configuration Parameters	The **DCCI CSCI** returns the attribute-value pairs (configuration parameters) to the INGST CSCI.
Return Status (PAN or PDRD)	The INGST CSCI returns the status of a request received from an **External Data Provider** by transmitting a Product Acceptance Notification (PAN) or a Product Delivery Record Discrepancy (PDRD) to the External Data Provider directly.

Table 4.2-2. INGST CSCI Interface Events (3 of 3)

Event	Interface Event Description
Store Raw Data	The raw data (L0) is provided from the **External Data Provider** to the SDPS. Some External Data Providers put this data on an external local disk for SDPS to pull while others push the data onto a local INGST CSCI internal disk.
Request Management Services	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document.

4.2.1.3 INGST Architecture

The Polling Ingest Interface (EcInPolling) polls accessible file system locations to detect data to be ingested. This process submits a Product Delivery Record (PDR) or the information for INGST to create a PDR. The Media Ingest Interface (EcInGUI) enables authorized science users or institutions to submit a PDR and the data to be ingested via physical media. The Cross-Mode Ingest Interface (EcInEmailGWServer) receives distribution notifications, via e-mail, and stores them as files in a location, which is polled. This process detects the notification files and creates Delivery Record files, which are put in a polling directory and detected by the Polling Ingest Interface.

The Polling Ingest Interface and the Media Ingest all submit ingest requests (containing PDRs) to the Ingest Request Manager (EcInReqMgr) and the EcInReqMgr submits data granule requests to the Ingest Granule Server (EcInGran). The EcInGran manages subsequent request processing. The EcInGran invokes a Data Transfer task to transfer data from external locations. The EcInGran also invokes a data pre-processing task to preprocess ingested data (e.g., process metadata and validate metadata parameters) and invokes the Data Server Insertion Task to insert data into the Data Server.

Figure 4.2-3 is the INGST CSCI architecture diagram. The diagram shows the events sent to the INGST CSCI processes and the events the INGST CSCI processes send to other processes.

Note: System startup and shutdown – Please refer to the release-related, current version of the Mission Operations Procedures for the EMD Project document (611) and the current EMD Project Training Material document (625), identified in Section 2.2.1 of this document.

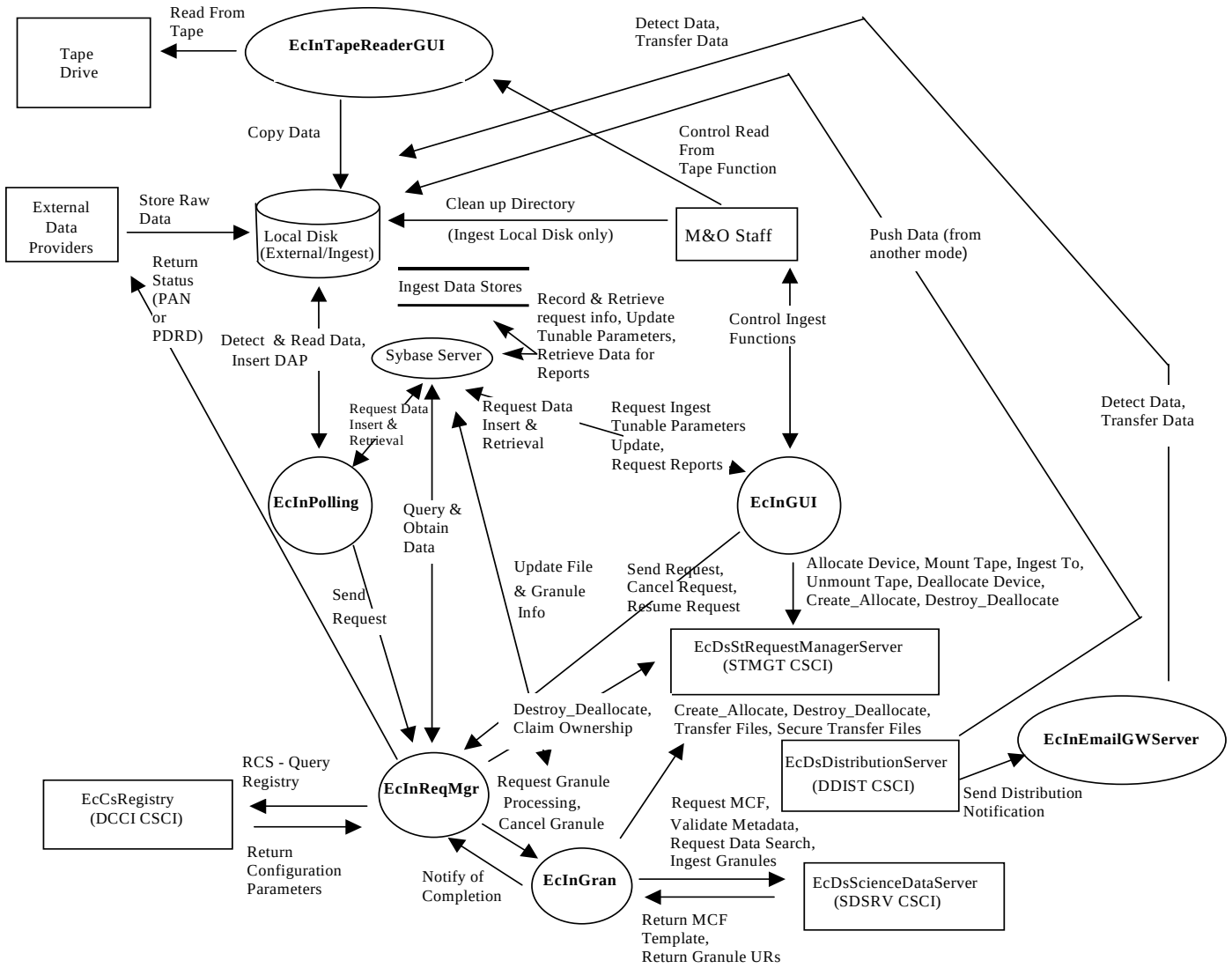


Figure 4.2-3. INGST CSCI Architecture Diagram INGST Process Descriptions

4.2.1.4 INGST Process Descriptions

Table 4.2-3 provides the descriptions of the processes shown in the INGST CSCI architecture diagram.

Table 4.2-3. INGST CSCI Processes (1 of 2)

Process	Type	Hardware CI	Source	Functionality
EcInPolling	Client	ICLHW	Developed	• Creates the appropriate polling request • Detects new files of interest at tunable periods of time in either external or local disk locations (by checking an agreed upon network location for available data) and can provide secure file transfer if data provider indicates secure file transfer by setting TransferFlag to 2 or 3 • Creates a unique identifier for the request • Submits requests • Reports the status of the ongoing requests
EcInGUI	GUI	ACMHW	Developed	Provides Maintenance and Operations (M&O) personnel the capability, via GUI Interface, • To perform physical media ingest (to ingest data from DTF media) • To monitor the ingest history log, to monitor the status of ongoing ingest requests, to cancel ingest requests and granules, and to resume suspended ingest requests and granules • To modify ingest configuration parameters
EcInReqMgr	Server	ICLHW	Developed	• Manages the ingest request traffic and the processing of the ingest requests, and • Provides the capability to process multiple ingest requests concurrently by placing the request in a queue • In the event of a failure, the EcInReqMgr process restores on-going requests from the Ingest database
EcInEmailGWServer	Server	INTHW	Developed	• Receives e-mail distribution notification messages • Stores e-mail messages into files • Detects new files of interest at a regular time interval, which can be configured, on a local disk • Creates a polling request and puts it on a local disk location
EclnTapeReaderGUI	GUI	ACMHW	Developed	• Reads data from physical media • Generates PDR(s) • Stores PDR(s) and data files in the PDR and data directoriesas specified in the corresponding configuration file

Table 4.2-3. INGST CSCI Processes (2 of 2)

Process	Type	Hardware CI	Source	Functionality
EcInGran	Server	ICLHW	Developed	Provides services to perform the required data preprocessing and the subsequent data insertion into the appropriate Data Server. The preprocessing of data consists of: • Converting the data (if needed) • Extracting the metadata into the standard SDPS metadata format (if needed) • Performing required metadata existence and parameter range checks • Updating the metadata with ingest specific metadata (e.g., start and stop date/time for ingest) EcInGran coordinates the ingest granule processing including: • Performing data preprocessing • Sending an insertion request to the appropriate Data Server • Updating the granule state • Transferring data files into Ingest • Building file lists • Grouping files with a valid ESDT
Sybase	Server	ICLHW	COTS	Stores and provides access to the INS internal data. In particular, the database stores the Ingest operations databases – Ingest History Logs and the Ingest request checkpoint state, and template information. See Section 4.2.1.6 INGST Data Stores.

EMD Baseline Information System (EBIS) Document 920-TDx-001 (HardwareDesign Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.2.1.5 INGST Process Interface Descriptions

Table 4.2-4 provides descriptions of the interface events shown in the INGST CSCI architecture diagram.

Table 4.2-4. INGST CSCI Process Interface Events (1 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Clean up Directory (Ingest local disk only)	One per Unix command to delete or remove	Directories on Ingest Local Disk	M&O Staff-/Unix command	The **M&O Staff** send clean up instructions to the Ingest Local Disk for data clean up. The Unix commands for delete and remove are executed from the Unix command line to clean up the Ingest Local Disk.
Detect Data	One per poll from EcInEmailGW Server	Directory on local disk	*Process:* EcInEmailGWServer *Class:* InEmailGWServer	The **EcInEmailGWServer** polls for notification files in an agreed upon location (on the Ingest Local Disk).
Transfer Data	Upon detection	Ingest Local Disk	*Process:* EcInEmailGWServer *Class:* InEmailGWServer	The **EcInEmailGWServer** sends distribution notification files and Delivery Record files to an agreed upon location (on the Ingest local disk).
Control Read From Tape Function	One per tape reader operation	*Process:* EcInTapeReaderGUI	M&O staff	The M&O staff controls the read from tape function by selecting the option on the GUI to read from tape.
Push Data (from another mode)	One per distribution	Ingest Local Disk	*Process:* EcDsDistributionServer *Class:* DsDdGranuleS	The **EcDsDistributionServer** pushes data, via the FTP Service, to the Ingest Local Disk when it is distributing data to be ingested.
Control Ingest Functions	One per Ingest Operation	*Process:* EcInGUI *Library:* InGuiUt *Classes:* InRequestControllerRPUtil, InMediaIngestRPUtil, InHistoryLogRPUtil, InOperatorToolsRPUtil	M&O staff	The **M&O staff** control the Ingest function by monitoring requests, canceling ingest requests and granules, resuming suspended ingest requests and granules, changing database parameters, viewing history, and performing manual media ingest via a GUI.
Record & Retrieve Request Info	One per request	Ingest Data Stores (database)	Sybase ASE (COTS)	Requests from the **EcInGUI** and **EcInPolling** processes are recorded into the Ingest database for reference and are a source for restarts and re-initializations of outstanding requests.

Table 4.2-4. INGST CSCI Process Interface Events (2 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Update Tunable Parameters	One per update of stored parameters	Sybase ASE (COTS)	*Process:* EcInGUI *Library:* InGuiUt *Class:* InOperatorToolsRPUtil	The EcInGUI sends requests to the **Sybase ASE** to update the stored parameters in the Ingest database affecting the functions of the EcInPolling, EcInReqMgr, and EcInGran processes.
Retrieve Data for Reports	One per data request	Sybase ASE (COTS)	*Process:* EcInGUI *Library:* InGuiUt Class: InHistoryLogRPUtil	The EcInGUI obtains data from the Ingest database via the **Sybase ASE** to generate reports.
Request Data Insert & Retrieval	One per request	Sybase ASE (COTS)	Operations Staff *Processes:* EcInGUI, EcInPolling *Library:* InDBaccUt *Classes:* InRequestProcessData, InRequestProcessHeader, InExternalDataProviderInfo, InNextAvailableID, InSystemParameters, InCurrentDataTypeMap, InDataTypeTemplate, InEDPAddressMap, InFileTypeTemplate, InMediaType, InRequestSummaryData, InRequestSummaryHeader, InValDataGranuleState, InValRequestState	Requests from the EcInGUI and EcInPolling processes are recorded into the Ingest database, via the **Sybase ASE**, for reference and are a source for restarts and re-initializations of outstanding requests.

Table 4.2-4. INGST CSCI Process Interface Events (3 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Ingest Tunable Parameters Update	One per update of stored parameters	Sybase ASE (COTS)	*Process:* EcInGUI *Library:* InGuiUt *Class:* InOperatorToolsRPUtil	The EcInGUI sends requests to the **Sybase ASE** to update the stored parameters affecting the functions of the EcInPolling, EcInReqMgr, and EcInGran processes.
Request Reports	One per data request	Sybase ASE (COTS)	*Process:* EcInGUI *Library:* InGuiUt Class: InHistoryLogRPUtil	The EcInGUI obtains data from the Ingest database to generate reports via the **Sybase ASE**.
Allocate Device	One allocation per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInGUI *Library:* InGuiUt *Class:* InMediaIngestRPUtil	The EcInGUI sends requests to the **EcDsStRequestManagerServer** to allocate peripheral devices for data ingesting.
Mount Tape	One per physical tape	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInGUI *Library:* InGuiUt *Class:* InMediaIngestRPUtil	The EcInGUI process sends requests to the **EcDsStRequestManagerServer** to load tapes to hardware peripherals for reading the tapes.
Ingest To	One data copy from peripheral device(s) per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInGUI *Library:* InGuiUt *Class:* InMediaIngestRPUtil	The EcInGUI sends requests to the **EcDsStRequestManagerServer** to copy files from peripheral resources to staging disk areas.

Table 4.2-4. INGST CSCI Process Interface Events (4 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Unmount Tape	One per physical tape	*Process:* EcDsStRequestManager Server *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInGUI *Library:* InGuiUt *Class:* InMediaIngestRPUtil	The EcInGUI process sends requests to the **EcDsStRequestManagerServer** to unload and detach tapes from hardware peripherals after reading or writing to the tapes.
Deallocate Device	One deallocation per request	*Process:* EcDsStRequestManager Server *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInGUI *Library:* InGuiUt *Class:* InMediaIngestRPUtil	The EcInGUI sends requests to the **EcDsStRequestManagerServer** to deallocate the previously allocated media resource.
Create_Allocate	One allocation per request	*Process:* EcDsStRequestManager Server *Library:* DsStRmClient *Class:* DsStRequestManager	***Process:*** EcInGUI *Library:* InResource *Class:* InResourceIF ***Process:*** EcInGran *Library:* InGranResource *Class:* InGranResourceIF	The EcInGUI and EcInGran processes send requests to the **EcDsStRequestManagerServer** to allocate areas on the local staging disk to store ingested data.

Table 4.2-4. INGST CSCI Process Interface Events (5 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Destroy_Deallocate	One deallocation per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	***Process:*** EcInGran *Library:* InGranResource *Class:* InGranResourceIF ***Processes:*** EcInGUI, EcInReqMgr *Library:* InResource *Class:* InResourceIF	The EcInGran, EcInGUI, and EcInReqMgr processes send requests to the **EcDsStRequestManagerServer** to deallocate a staging disk area (to remove an existing staging disk area from usage).
Claim Ownership	One per staging disk area	*Process:* EcDsStRequestManager Server *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInReqMgr *Library:* InResource *Class:* InResourceIF	The EcInReqMgr sends requests to the **EcDsStRequestManagerServer** to claim ownership of (take responsibility for deallocating) an existing staging disk area.
Send Request	One per request to Request Manager	*Process:* EcInReqMgr *Class:* InRequestManager	*Processes:* EcInPolling, EcInGUI *Library:* InGuiUt *Classes:* InPollingIngestSession, InMediaIngestRPUtil	Processing requests from one of the two ingest processes (**EcInPolling** and **EcInGUI**) are sent to the EcInReqMgr.
Cancel Request	One per Ingest Request/Granule	*Process:* EcInReqMgr *Class:* InRequestManager	*Process:* EcInGUI *Library:* InGuiUt *Class:* InRequestControllerRPUtil	The **EcInGUI** is the interface to the EcInReqMgr process to cancel a request or one of its granules.

Table 4.2-4. INGST CSCI Process Interface Events (6 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Resume Request	One per suspended Ingest Request/Granule	*Process:* EcInReqMgr *Class:* InRequestManager	*Process:* EcInGUI *Library:* InGuiUt *Class:* InRequestControllerRPUtil	The **EcInGUI** is the interface to the EcInReqMgr process to resume a suspended request or one of its granules.
Update File & Granule Info	One per file or granule update	Sybase ASE (COTS)	*Process:* EcInGran *Library:* InDBaccUt *Classes:* InRequestProcessData, InRequestFileInfo	The EcInGran process sends requests to the **Sybase ASE** to update file and granule information obtained from the Ingest database.
Request Granule Processing	One per granule processing request	*Process:* EcInGran *Class:* InGranuleAsync_S	*Process:* EcInReqMgr *Class:* InRequest	The EcInReqMgr sends processing requests to the **EcInGran** process for granule processing.
Cancel Granule	One per Granule	*Process:* EcInGran *Class:* InGranuleAsync_C	*Process:* EcInReqMgr *Class:* InRequest	The EcInReqMgr sends a cancel message to the **EcInGran** process.
Notify of Completion	One per granule completion	*Process:* EcInReqMgr *Library:* InGranuleC *Class:* InGranuleAsync_C	*Process:* EcInGran *Class:* InGranuleAsync_S	The EcInGran process sends a completion notification to the **EcInReqMgr** when a granule for a request is completed.
Transfer Files	One per Science Data file activity	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInGran *Library:* InGranResource *Class:* InGranResourceIF	The EcInGran sends requests to the **EcDsStRequestManagerServer** to transfer (copy) data files to a staging disk.
Secure Transfer Files	One per request	*Process:* EcDsStRequestManagerServer *Library:* DsStRmClient *Class:* DsStRequestManager	*Process:* EcInGran *Library:* InGranResource *Class:* InGranResourceIF	The EcInGran sends requests to the **EcDsStRequestManagerServer** to secure transfer (scp) data files to a staging disk.

Table 4.2-4. INGST CSCI Process Interface Events (7 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Request MCF	One per access of MCF	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	The EcInGran process requests the MCF template from the **EcDsScienceDataServer**.
Validate Metadata	One per metadata validation	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	The EcInGran process requests the **EcDsScienceDataServer** to perform a validation of the metadata files.
Request Data Search	One per granule pointer in linkage file	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClQuery	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	The EcInGran process sends a search request to the **EcDsScienceDataServer** for a granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Ingest Granules	One per insert into the archive	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClRequest	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataServerInsertionTask	The EcInGran process requests a file or files to be inserted into the SDSRV (**EcDsScienceDataServer**) inventory and archives, and the associated metadata is catalogued in the SDSRV inventory (archives), as a granule of a particular ESDT short name and version.
Return MCF Template	One per set of external data received by EMD	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	The EcInGran process receives the MCF information as part of the GetMCF service call from the **EcDsScienceDataServer**.

Table 4.2-4. INGST CSCI Process Interface Events (8 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Granule URs	One per input pointer in metadata or per granule pointer in linkage file	*Process:* EcInGran *Library:* InPreprocess *Class:* InDataPreprocessTask	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The EcInGran process receives the granule URs for the granules requested in the data search from the **EcDsScienceDataServer**.
Send Distribution Notification	One per distribution request	*Process:* EcInEmailGWServer *Classes:* InEmailGWServer, InEmailParser	*Process:* EcDsDistributionServer *Library:* DsDdSSh *Classes:* DsDdMedia, DsDdMediaDist	The EcInEmailGWServer receives distribution notifications via e-mail from the **EcDsDistributionServer**.
Return Configuration Parameters	One set per request	*Process:* EcInReqMgr *Library:* InResource *Class:* InResourceIF	*Process:* EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	The **EcCsRegistry** returns the attribute-value pairs (configuration parameters) to the EcInReqMgr.

Table 4.2-4. INGST CSCI Process Interface Events (9 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Communications Support	One service per request	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Library:*** event *Class:* EcLgErrorMsg ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Processes:* EcInReqMgr, EcInGran, EcInPolling, EcInGUI, EcInEmailGWServer	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS process. The process services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • File Transfer Services • Network & Distributed File Services • Bulk Data Transfer Services • Name/Address Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry

Table 4.2-4. INGST CSCI Process Interface Events (10 of 10)

Event	Event Frequency	Interface	Initiated By	Event Description
Query & Obtain Data	One per query of Ingest Database	Sybase ASE (COTS)	*Process:* EcInReqMgr *Library:* InDBaccUt *Classes:* InDataTypeTemplate, InRequestFileInfo, InRequestProcessData, InRequestProcessHeader, InCurrentDataTypeMap, InEDPAddressMap, InExternalDataProviderInfo, InGranuleServerInfo, InSystemParameters	The EcInReqMgr sends requests to the **Sybase ASE** for ingest processing information from the Ingest database.
Return Status (PAN or PDRD)	One per request	External Data Provider	*Process:* EcInReqMgr *Class:* InRequest	The EcInReqMgr returns status of a request to the **External Data Providers** via a Product Acceptance Notification (PAN) or a Product Delivery Record Discrepancy (PDRD).
Detect and Read Data	One per poll from EcInPolling	Various ID directories on various Ingest Local Disks	*Process:* EcInPolling *Class:* InPollingIngestSession	The EcInPolling polls for data files or Delivery Record files in an agreed location (on the **Ingest Local Disk**).
Insert DAP	One per request	Ingest Local disk	*Process:* EcInPolling *Class:* InPollingIngestSession	Delivered Algorithm Packages (DAPs) are stored on the **Local Disk** for insertion into the SDPS with the Ingest Polling process.
Store Raw Data	One per data delivery	Ingest Local disk	External Data Providers	The **External Data Providers** send raw data to the EMD Ingest local disk via the FTP service.
Read From Tape	One per tape reader operation	Tape Drive	Process: EcInTapeReaderGUI	The EcInTapeReaderGUI reads the data from tape to copy to disk.
Copy Data	One per tar file on the tape	EcInTapeReaderGUI	System Call (a TAR command)	The EcInTapeReaderGUI copies data read from tape to disk.

4.2.1.6 INGST Data Stores

The INGST CSCI uses the COTS product Sybase to store related INGST information on a physical medium. The stored information is divided into four functional areas:

1. Checkpoint and reactivate ingest processing
2. Summary or historical information for collecting and reporting metrics
3. Ingest configuration (e.g., thresholds) and template information
4. Validation tables for the INS GUI and software

Table 4.2-5 provides descriptions of the individual data stores used by the INGST CSCI. The architecture diagram shows a single data store entitled "Ingest Data Stores" for simplification.

Table 4.2-5. INGST CSCI Data Stores (1 of 2)

Data Store	Type	Description
InRequestProcessHeader	Sybase	Provides checkpoint storage of ingest request processing information associated with a given ingest request. Upon request process completion, copies of these records are stored in InRequestSummaryHeader data store and the request processing ingest information is deleted.
InRequestProcessData	Sybase	Provides checkpoint storage of data granule processing information associated with a given ingest request. Upon request process completion, copies of these records are stored in the InRequestSummaryData data store and the granule processing data information is deleted.
InRequestFileInfo	Sybase	Provides checkpoint storage of file information associated with a data granule within a given ingest request.
InRequestSummaryHeader	Sybase	Provides long-term storage of summary request-level statistics associated with a given ingest request. Summary records are copied upon ingest request processing completion and the processing records are deleted from the system.
InRequestSummaryData	Sybase	Provides long-term storage of summary data type statistics associated with a given data granule in a given ingest request. Summary records are copied upon ingest request processing completion and the processing records are deleted from the system.
InSourceMCF	Sybase	Initially, pre-populated with the valid metadata types for each file type. It "points" to the metadata and indicates "how" to handle the data in a standard object description language (ODL) format.
InSystemParameters	Sybase	Stores current system thresholds that limit ingest request traffic and data volume.
InExternalDataProviderInfo	Sybase	Stores thresholds on ingest request traffic and data volume for External Data Providers.
InValGranuleServerUR	Sybase	Provides the name of each configured Granule Server.
InGranuleServerInfo	Sybase	Stores thresholds on granule traffic and data volume for each Granule Server.
InDataTypeTemplate	Sybase	Initially, pre-populated with current, valid Earth Science Data Types (ESDTs) that the INS is capable of ingesting.

Table 4.2-5. INGST CSCI Data Stores (2 of 2)

Data Store	Type	Description
InGranuleQueue	Sybase	Stores the granule server queues.
InFileTypeTemplate	Sybase	Initially, pre-populated with all valid file types that make up an ESDT.
InMediaType	Sybase	Stores the valid values of the media types available that can be ingested.
InNextAvailableID	Sybase	Stores the next available RequestID to be given.
InValDataGranuleState	Sybase	Stores all the valid values for a data granule state.
InValIngestType	Sybase	Stores all the valid values for an ingest type.
InValNotifyType	Sybase	Stores all the valid values for a notify type.
InValParameterClass	Sybase	Initially, pre-populated with all the valid values for a parameter class.
InValRequestState	Sybase	Stores all the valid values for a request state.
InEDPAddressMap	Sybase	Initially, pre-populated with the IP address for an External Data Provider.
InCurrentDataTypeMap	Sybase	Initially pre-populated with current valid Earth Science Data Types (ESDTs) Ingest is capable of ingesting and the current Ingest version id for each one.
InSSHCipherMap	Sybase	Initially, pre-populated with all supported secure copy cipher type and its corresponding TransferFlag configuration.
InValFileCksumType	Sybase	Initially, pre-populated with all supported file checksum types.

4.2.2 Ingest Subsystem Hardware

4.2.2.1 Ingest Client Hardware CI Description

The Ingest Client HWCI (ICLHW) Server accommodates the required ingest volumes, including I/O, and processing capabilities to support internal data transfers associated with metadata validation and extraction, and to transfer data to the Data Server or Data Processing Subsystem. The disks are sized to accommodate the functionality and provide contingency space for the transfer of more than one day's worth of data within a 24-hour period.

Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the Ingest HWCI and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.2.2.2 Ingest Workstation Description

The Ingest Workstation is provided to execute the Ingest GUI. This workstation enables the operator to remotely monitor the Ingest Servers and the Ingest processes, from media ingest to remote ingest.

Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the Ingest HWCI and document 920-TDx-002 (hardware-Software Map) provides site-specific hardware/software mapping.

4.3 Client Subsystem Overview

The Client Subsystem (CLS) is a set of CSCIs and processes that provide EMD end-user services.

These services include allowing users to:

- Submit ASTER Data Acquisition Requests (DARs)
- Create, retrieve and update ECS user profiles
- View HDF formatted files

In addition, the workstations operating within an ECS CLS contains infrastructure support software as part of the CSS and platform support software.

Client Subsystem Context

Figure 4.3-1 is the Client Subsystem context diagrams. The diagrams show the events sent to the CLS and the events the CLS sends to other SDPS or CSMS subsystems.

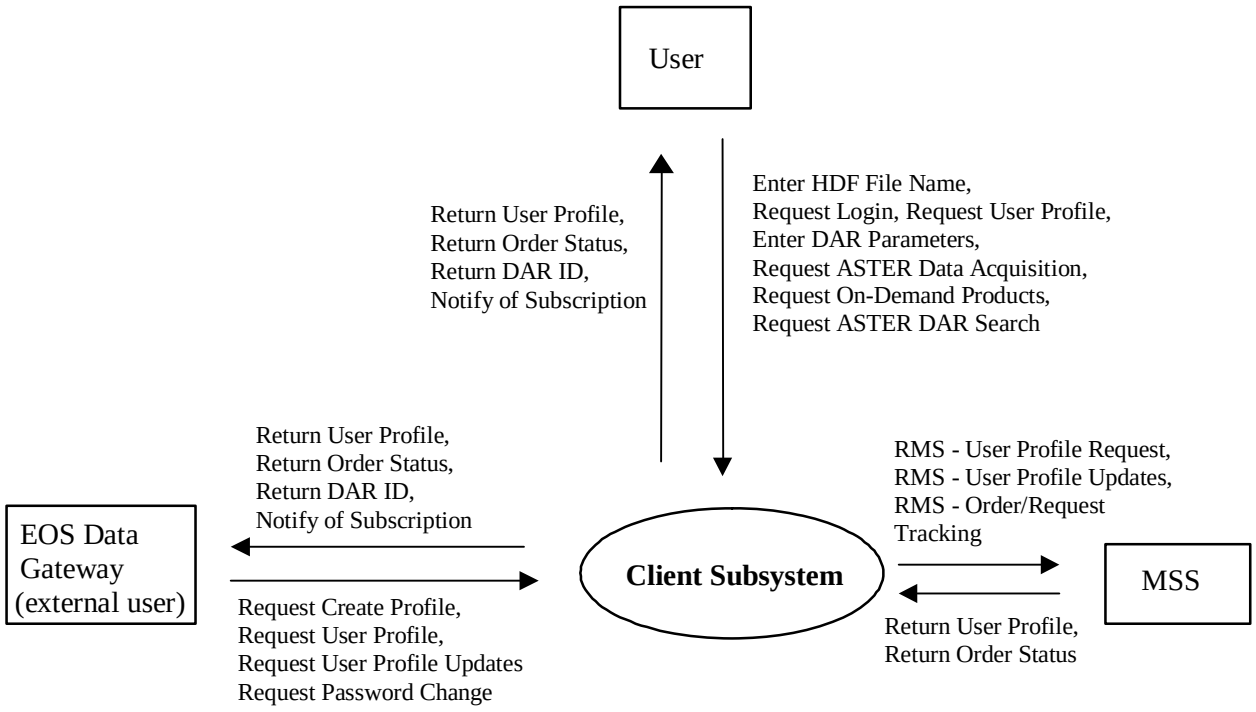


Figure 4.3-1. Client Subsystem Context Diagram

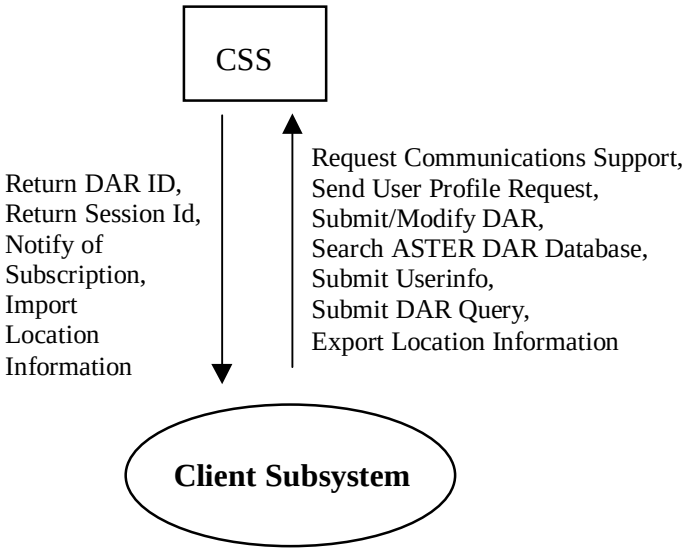


Figure 4.3-1. Client Subsystem Context Diagram (cont.)

The EOS Data Gateway, the Version 0 Client (Web version) performs the data searches and retrievals. The EOS Data Gateway is treated as an external entity since the design and design documentation is controlled under another contract. Documentation on the V0 Web Client (EOS Data Gateway) design can be accessed through the Universal Resource Locator http://harp.gsfc.nasa.gov/~imswww/pub/manuals/imsdesign.html.

Table 4.3-1 provides descriptions of the interface events shown in the Client Subsystem context diagram.

Table 4.3-1. Client Subsystem Interface Events (1 of 4)

Event	Interface Event Description
Enter HDF File Name	This is a file name for a Hierarchical Data Format (HDF) file. The **user** opens the file to see the data in the file.
Request Login	A user name and password for the **user** are provided for access to the ECS. The user name and password are sent to the CSS (via a request for communications support).
Enter DAR Parameters	The **user** enters parameters (as specified in the Interface Control Document (ICD)) required for submittal or modification of Data Acquisition Requests (DARs) in accordance with the ASTER GDS Interface Requirements Document (IRD). Upon completion of the selection or modification of DAR parameters, the user may submit a DAR. In addition, the user may specify DAR parameters for a search of the DAR database.

Table 4.3-1. Client Subsystem Interface Events (2 of 4)

Event	Interface Event Description
Request ASTER Data Acquisition	A **user** submits a request (to the CLS) to have ASTER data taken (a data acquisition request or DAR) using the parameters entered into the Java DAR Tool. DAR parameters are required for submittal of DARs as specified in the ASTER GDS IRD/ICD. As the result of a successfully submitted DAR, the user receives a DAR ID. This is a string of characters used to track a DAR. The user receives notification every time data resulting from this DAR is received by the ECS.
Request ASTER DAR Search	A **user** submits a request (to the CLS) to search the ASTER DAR database by DAR parameters or a specific DAR ID to determine if a scene of interest (to the user) has been acquired by the ASTER instrument.
Request Management Services	The **MSS** provides a basic management library of services to the subsystems, implemented as client or server applications, using the CSS Process Framework. The basic management library of services includes: • **System startup and shutdown** - Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document. • **Create Profile Request** - The **MSS** receives user information for becoming a registered user of the ECS from the CLS. The MSS sends a response to the user when the request is received. • **User Profile Request** - The **MSS** provides requesting subsystems with User Profile parameters such as e-mail address and shipping address upon request by authorized users to support their processing activities. • **Order/Request Tracking** - The CLS uses CGI scripts to interface with the **MSS** Order/Request Tracking service to create a user product order and submit the order to the PLS. • **User Profile Updates** - The **MSS** receives user profile parameter updates from a user and makes the updates in the user profile database. • **Password Change Request** - The CLS sends requests on behalf of ECS users to the **MSS** to change users' authenticators in the MSS database.
Return User Profile	The user profile is returned from the **MSS** to the CLS to be returned to the external user via the EOS Data Gateway.
Return Order Status	The CLS receives an order id and status for the requested ECS product from the **MSS** and returns the order id and status to the ECS or V0 IMS (via the EDG) user (to track the order).
Request Create Profile	Users submit a request to be a registered user of the ECS. Registered users can be given special privileges not awarded to guests, such as the capability to order data on a media at a cost. The user request is sent through the **EOS Data Gateway** (**EDG**), which creates both an EDG and an ECS User Profile. The user enters his or her addresses (user, shipping, billing and e-mail) and other important information. This profile information is used to establish an EDG profile and is forwarded through the CLS to the MSS. The MSS creates the ECS profile.

Table 4.3-1. Client Subsystem Interface Events (3 of 4)

Event	Interface Event Description
Request User Profile	A User Profile Request is a search for a User Profile from the user via the **workbench** or **EDG**. In response, the CLS receives the user profile, which contains information about a user that must be maintained. This includes, but is not limited to, mailing, billing, and shipping addresses, phone number, electronic mail address, project account number and project organization.
Request User Profile Updates	The user can update their User Profile information through the **EOS Data Gateway**. This includes their addresses (user, shipping, billing, and e-mail) and other important information. This updated profile information (profile2.odl) is forwarded through the CLS to the MSS. The EOS Data Gateway uses the profile2.odl file rather than a live interface with the CLS as its source of user information.
Request Password Change	The user can request a change of his/her EDG User Profile passwords through the **EOS Data Gateway**. The user enters his current and new passwords. This information is used to change the EDG password and the passwords (old and new) are forwarded through the CLS to the MSS. The MSS changes the ECS password.
Return DAR ID	As the result of a successfully submitted DAR, the user receives a DAR ID from the **CSS** via the CLS or EDG. This is a string of characters used to track a DAR.
Notify of Subscription	The user receives notification every time data resulting from a successfully submitted DAR is received by the ECS from the **CSS** via the CLS or EDG.
Request Communications Support	The **CSS** provides a library of services available to each SDPS and CSMS subsystem. The subsystem services required to perform specific assignments are requested from the CSS. These services include: • CCS Middleware Support • Database Connection Services • Name/Address Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event logging • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry • Request Distribution Media Options from the Configuration Registry
Send User Profile Request	The CLS sends a user profile request to the **CSS** to communicate with the ASTER GDS.
Submit/Modify DAR	The user submits a DAR after selecting or modifying DAR parameters to the **CSS**. As the result of a DAR submission, the user receives a DAR ID. This is a string of characters used to track a DAR. The user receives notification every time data resulting from this DAR is received by the system.
Search ASTER DAR Database	The CLS submits a request to the **CSS** to search the ASTER GDS DAR database for DARs and their respective status (i.e., acquired scenes). Search qualifications may be in the form of DAR parameters or DAR Ids. To get a status of the search, users may view the Search Status via the Java DAR Tool.
Submit Userinfo	User name and password are sent to the **CSS**.
Submit DAR Query	The CLS sends the parameters required for querying DARs to the **CSS** as one of the following three queries: queryxARContents, queryxARScenes, or queryxARSummary. The results of the query are returned to the CLS.

Table 4.3-1. Client Subsystem Interface Events (4 of 4)

Event	Interface Event Description
Export Location Information	The CCS Middleware CSC stores physical and logical location information received from DPS in the **CSS** (CCS NameServer).
Return Session Id	The **CSS** returns a session id to the CLS for the user to communicate with the ASTER GDS.
Import Location Information	The CLS requests server location information from the **CSS** (CCS NameServer).

Client Subsystem Structure

The Client Subsystem is three CSCIs:

- The Workbench (WKBCH) CSCI includes the set of ECS applications and libraries that provide access to the ECS data and services. There are two tools: the EOSView and the Java DAR Tool (JDT). The EOSView is an X/Motif application resident on a science user's workstation. The Java DAR Tool is a java-based web application that can be accessed through a web browser.
- The Desktop (DESKT) CSCI provides the User Profile Gateway server to communicate with the MSS User Registration Server for creating new user accounts, obtaining user profile information to update user information.

The CLS contains no HWCIs. The DMS hardware (Interface Server) provides the processing and storage for the WKBCH software. In addition, the User Profile Gateway Server is used to create, retrieve and update user profiles for ECS users via the EOS Data Gateway. The CLS is required to support the following hosts: SGI IRIX, and SUN Solaris 8. Currently, the ECS only supports Personal Computers running Windows 95 and higher versions of windows and SUN Solaris running with Netscape Navigator versions of 4.7 and higher versions.

The Interface Server is documented in 920-TDx-001 which provides descriptions of the Interface Server HWCI.The Workbench software executed on these hosts enables user access to the ECS data and services. The Interface Server also provides storage for user session data and the Java DAR Tool (JDT) map data. Detailed mappings can be found in the site-specific hardware/software mapping in baseline document number 920-TDx-002.

Use of COTS in the Client Subsystem

- Netscape Navigator

  The Netscape Navigator Web browser accesses the CLS inside a DAAC. The users can use the Web browsers they already have at their facilities.

- Sun One Web Server

  The Sun One Web Server is used to serve the Web pages for the Java DAR Tool.

- Providence Software Solution's DSC

  The Development Solution for the C programming Language (DSC) is used as a widget set and development tool for the EOSView application of the WKBCH CSCI.

- Interactive Data Language (IDL)

  IDL is used by EOSView to provide the visualization features for users.

- CCS Middleware Client

  CCS Middleware Client provides CLS with communications between other subsystems. CCS Middleware can reside on one or both sides of the interface. An instance must be installed on the platform where CLS resides. Although the CCS Middleware Client is part of CSS, this COTS product must be installed for CLS to run in the SDPS operational and test environment. The Java DAR Tool does not require the use of CCS Middleware.

Error Handling and processing

EcUtStatus is a class used throughout the ECS custom code for general error reporting. It is almost always used as a return value for functions and allows detailed error codes to be passed back up function stacks.

The WKBCH CSCI contains the EOSView and Java DAR Tool (JDT) standalone COTS products. EOSView has its own custom error handling.

The JDT uses the Java exception facilities to check error conditions. Exceptions are either handled in the class that produced the error or they are sent back to the calling method/class. At some point in the calling chain, the exception is either handled by the application code or it is dumped out in the stack trace (which appears in the Java Console if it's a client-side exception or in the jess.log if it's a server-side exception). The JDT's exception classes are organized in an exceptions package (jdt.aux.exceptions).

The ODFRM CSCI has no special error handling. ODFRM is a cgi application, which uses EcUtStatus to communicate errors. Functions can return an EcUtStatus object, indicating success or failure.

The DESKT CSCI consists of the User Profile Gateway. The User Profile Gateway uses EcUtStatus and exceptions. Functions can return an EcUtStatus object, indicating success or failure or throw an exception. The User Profile Gateway does not have any special error processing.

4.3.1 Workbench Computer Software Configuration Item Description

4.3.1.1 Workbench Functional Overview

The Workbench (WKBCH) CSCI is a set of application programs, which implement the core functionality of the CLS science user interface. The V0 Client performs the data search and retrieval. The EOS Data Gateway is treated as an external entity since the design and design documentation is controlled under another contract. The WKBCH CSCI provides users the

capability to submit data acquisition requests for the ASTER instrument data, and to see data products in HDF format.

The Java DAR Tool handles user requests to acquire ASTER data from the satellite. The user specifies when and where the data is to be taken. The DAR is submitted to the ASTER GDS in Japan for review. The acceptance of the DAR by Japan is not immediate (though the acknowledgment of receipt is) and the Java DAR Tool cannot determine the DAR status immediately. In addition to the submission and modification of acquisition requests, the Java DAR Tool allows users to search the ASTER DAR database (via MOJO and the DAR Communications Gateway) in order to examine the status of DARs (i.e., the number and quality of acquired scenes). Search for DARs on a parameter base or by DAR Id. The Java DAR Tool is a java-based web tool that runs on Sun workstations and PCs.

EOSView is an HDF-EOS viewer that enables users to visualize data they receive from the ECS. EOSView can take any HDF-EOS data file and perform basic visualization functions on it. EOSView is not meant to provide sophisticated data analysis functions like those found in COTS products such as IDL. EOSView is a GUI application used on UNIX platforms using X/Windows and Motif.

4.3.1.2 Workbench Context

Figure 4.3-2 is the WKBCH CSCI context diagram. The diagram shows the events sent by WKBCH to other SDPS and CSMS subsystems and events sent to WKBCH from the other SDPS and CSMS subsystems.

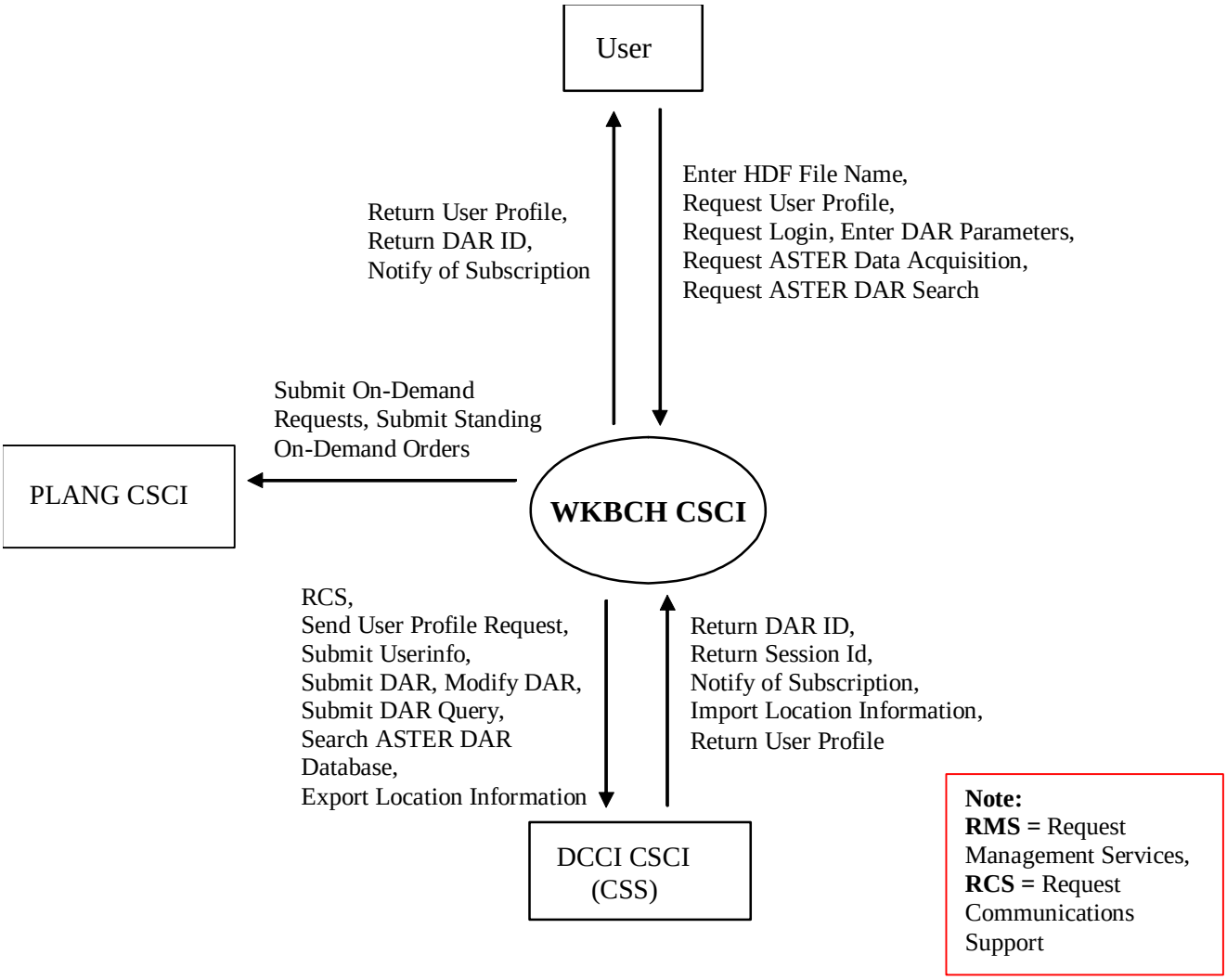


Figure 4.3-2. WKBCH CSCI Context Diagram

Table 4.3-2 provides descriptions of the interface events shown in the WKBCH CSCI Context Diagram.

Table 4.3-2. WKBCH CSCI Interface Events (1 of 3)

Event	Interface Event Description
Enter HDF File Name	This is the name of an HDF file to be opened in EOSView. A file name passed to EOSView enables EOSView to present the file contents to the **user**. This file can be obtained through any means (i.e., from a search, browse, acquire, or from a friend). The EOSView enables the user to manipulate an HDF file into multiple types of displays. Images, metadata, and actual data values can be viewed. Multiple images can be animated.

Table 4.3-2. WKBCH CSCI Interface Events (2 of 3)

Event	Interface Event Description
Request User Profile	A User Profile Request is a search for a User Profile from the **user**. There are two methods used, one is by an encrypted user name and password and the other is by user identification (ID). In response, the WKBCH CSCI receives the user profile, which contains information about a user that must be maintained. This includes mailing, billing, and shipping addresses, phone number, electronic mail address, etc.
Request Login	Enter the user name and password that identifies the user to the ECS. The WKBCH CSCI receives the user name and password and requests a user profile from the MCI for the **user**.
Enter DAR Parameters	The **user** enters parameters (as specified in the Interface Control Document (ICD)) required for submittal or modification of Data Acquisition Requests (DARs) in accordance with the ASTER GDS Interface Requirements Document (IRD). Upon completion of the selection or modification of DAR parameters, the user may submit a DAR. In addition, the user may specify DAR parameters for a search of the DAR database.
Request ASTER Data acquisition	A **user** submits a request (to the CLS) to have ASTER data taken (a data acquisition request or DAR) using the parameters entered into the Java DAR Tool. DAR parameters are required for submittal of DARs as specified in the ASTER GDS IRD/ICD. As the result of a successfully submitted DAR, the user receives a DAR ID. This is a string of characters used to track a DAR. The user receives notification every time data resulting from this DAR is received by the ECS.
Request ASTER DAR Search	A **user** submits a request (to the CLS) to search the ASTER DAR database by DAR parameters or a specific DAR ID to determine if a scene of interest (to the user) has been acquired by the ASTER instrument.
Return DAR ID	As the result of a successfully submitted DAR, the **DCCI CSCI** sends a DAR ID to the WKBCH CSCI. The **user** receives a DAR ID from the WKBCH CSCI. This is a string of characters used to track a DAR.
Return Session Id	The **DCCI CSCI** returns a session id to the WKBCH CSCI for the user to communicate with the ASTER GDS.
Notify of Subscription	The **DCCI CSCI** sends notification to the WKBCH CSCI when the data for a subscription arrives in the ECS. The **user** receives notification every time data resulting from a successfully submitted DAR is received by the ECS from the DCCI CSCI via the WKBCH CSCI.
Import Location Information	The WKBCH CSCI receives physical and logical server location information from the **DCCI CSCI**.
Return User Profile	The **DCCI CSCI** returns a user profile based upon a request from the user. The **user** receives the profile information from the WKBCH CSCI.

Table 4.3-2. WKBCH CSCI Interface Events (3 of 3)

Event	Interface Event Description
Request Communications Support	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry – Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Send User Profile Request	The WKBCH CSCI sends a user profile request to the **DCCI CSCI** to communicate with the ASTER GDS.
Submit Userinfo	The user name and password are sent to the **DCCI CSCI**. The DCCI CSCI sends back a session id.
Submit DAR	The WKBCH CSCI submits a request to the **DCCS CSCI** to have ASTER data taken (a data acquisition request or DAR) using the parameters entered into the Java DAR Tool. DAR parameters are required for submittal of DARs as specified in the ASTER GDS IRD/ICD. As the result of a successfully submitted DAR, the user receives a DAR ID. This is a string of characters used to track a DAR. The user receives notification every time data resulting from this DAR is received by the ECS.
Modify DAR	The WKBCH CSCI sends the modified DAR parameters to the **DCCI CSCI** to submit a DAR to the ASTER GDS.
Submit DAR Query	The WKBCH CSCI sends the parameters required for querying DARs to the **DCCI CSCI** as one of the following three queries: queryxARContents, queryxARScenes, or queryxARSummary. The results of the query are returned to the WKBCH CSCI.
Search ASTER DAR database	A user submits a request to the **DCCI CSCI** to search the ASTER GDS DAR database for DARs and their respective status (i.e., acquired scenes). Search qualifications may be in the form of DAR parameters or DAR Ids. To get a status of the search, users may view the Search Status via the Java DAR Tool.
Export Location Information	The WKBCH CSCI sends the server physical and logical location information to the **DCCI CSCI**.
Request Management Services	The **MCI** provides a basic management library of services to the subsystems, implemented as client or server applications, using the DCCI CSCI Process Framework. The basic management library of services includes: • **System startup and shutdown –** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.

4.3.1.3 Workbench Architecture

EOSView and the DAR Tool run on the user's workstation. Since these tools have no interfaces to each other and have distinct operations, their uses are described separately. Figure 4.3-3 is the EOSView architecture diagram. The diagram shows the events sent to the EOSView tool.

Since EOSView is a stand-alone application, it has no interfaces and gets initialized from the command line of a Unix platform. EOSView allows the user to provide a HDF file name to view HDF formatted files.

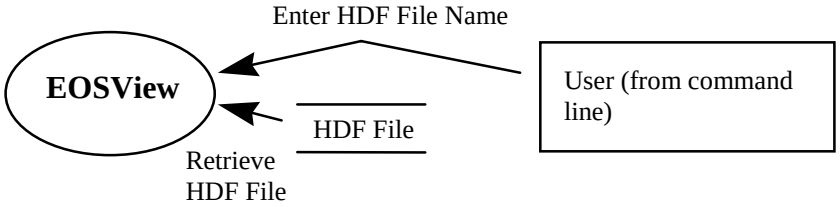


Figure 4.3-3. EOSView Architecture Diagram

Figure 4.3-4 is the Java DAR Tool architecture diagram. The diagram shows the events sent to the JDT process and the events the JDT process sends to other CSCIs, CSCs, or processes.

The Java DAR Tool is initiated from the web browser as an applet and can be initiated from the command line as a Java application. The Java DAR Tool uses the CSS MOJO Gateway as the gateway to all ECS services. JDT submits DARs to the CSS MOJO Gateway, which in turn submits them to the DAR Communications Gateway CSC. DAR requests can be a DAR submission or a modification to an existing DAR. After the DAR is successfully submitted, the Java DAR Tool submits a subscription on behalf of the user to get notification when the data associated with the DAR is ingested into the ECS. The Java DAR Tool retrieves the configuration parameters from the jdt.cfg file (ShortName, VersionId, Action) to determine the parameters to send to the Subscription Server.

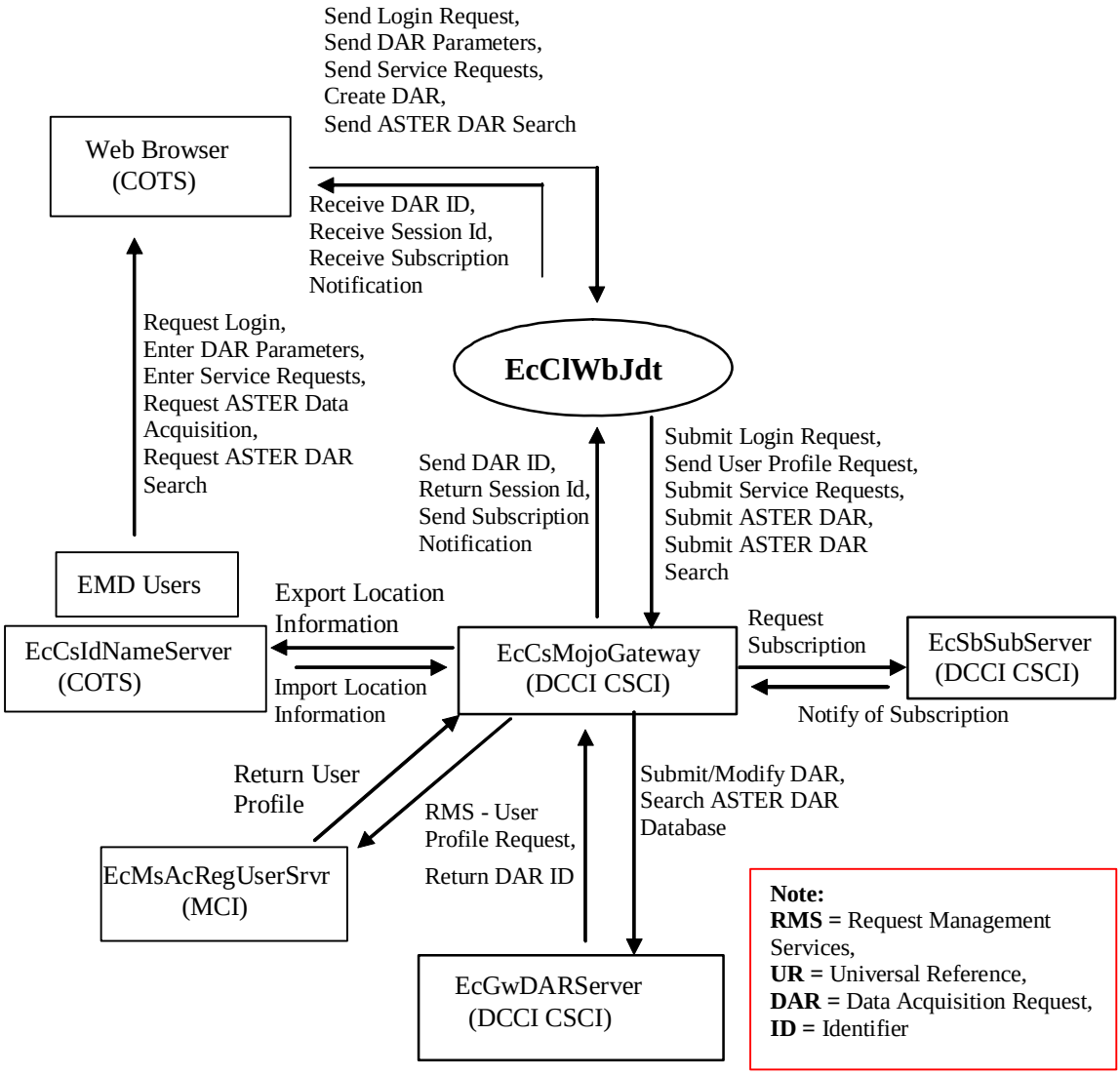


Figure 4.3-4. Java DAR Tool Architecture Diagram

4.3.1.4 Workbench Process Descriptions

Table 4.3-3 provides descriptions of the processes shown in the EOSView and DAR Tool Architecture Diagrams.

Table 4.3-3. WKBCH CSCI Processes

Process	Type	Hardware CI	COTS/ Developed	Functionality
EOSView	GUI	INTHW	Developed	This is the end user tool that provides data visualization functions for the ECS data. It is used by general users of the system and personnel within the DAAC and SCF for Quality Assurance (QA) checks of products.
EcClWbJdt	GUI	INTHW	Developed	This is a Java GUI that enables users to submit DARs to the ASTER GDS through the CSS DAR Communications Gateway (via CSS MOJO Gateway). When the DAR is submitted, a DAR Identifier is returned to the user. A subscription is submitted on behalf of the user, asking for notification whenever a data granule with the specified DAR ID is inserted into the SDPS archives. The Java DAR Tool also has the capability to modify DARs in accordance with the ASTER ICD. In addition, the Java DAR Tool allows the user to search for and retrieve DARs and their status (i.e., the scenes acquired).
EcClDtUserProfileGateway	Server	INTHW	Developed	The User Profile Gateway is a retrieval engine for users via the EOS Data Gateway. The server listens for calls on a socket from the EOS Data Gateway. Server Supports: • Multiple concurrent requests

In the EMD Baseline Information System (EBIS) Document 920-TDx-001 (HardwareDesign Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.3.1.5 Workbench Process Interface Descriptions

Table 4.3-4 provides descriptions of the process interface events shown in the EOSView and Java DAR Tool architecture diagrams.

Table 4.3-4. WKBCH CSCI (EOSView) Process Interface Events (1 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Enter HDF File Name	One per user identified file	Read from command line.	User *Process:* EOSView (COTS)	The user types a file name on the **EOSView** user interface to tell the GUI which file to open.
Retrieve HDF File	One file per request	File name provided by the user as read from the script file that captured it while active.	*Process:* EOSView (COTS)	The **EOSView** process retrieves the HDF File from a data store based on the file name provided by the user.

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (2 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Submit Login Request	One per User	*Process:* EcCsMojoGateway *Class:* EcMjDCELoginProxy	*Process:* EcClWbJdt Class: ClWbUrUserInfo	The EcClWbJdt sends the user name and password to the **EcCsMojoGateway** for use of ECS data and services.
Request User Profile	One per request	*Process:* EcCsMojoGateway *Class:* EcMjRetrieveProfileProxy	*Process:* EcClWbJdt *Class:* JDTApplet	The EcClWbJdt sends user profile requests to the **EcCsMojoGateway** to get user profile information for DAR submit authorization based upon the user id information provided by the user.
Submit Service Requests	Per user request	*Process:* EcCsMojoGateway Library: EcCsMojoGateway *Classes:* EcMjDarSubmitDarProxy, EcMjDarModifyDarProxy, EcCsRetrieveProfileProxy, EcMjDarQueryxARScenes Proxy, EcMjECSSbsrvProxy	*Process:* EcClWbJdt *Class:* JDTApplet	The EcClWbJdt accepts the requests of the user and submits the requests to the **EcCsMojoGateway**, which in turn sends the requests to the EcSbSubServer, the EcGwDARServer, or the EcMsAcRegUserSrvr.

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (3 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Submit ASTER DAR	One per request	*Process:* EcCsMojoGateway *Classes:* EcMjDarSubmitDarProxy, EcMjDarModifyDarProxy	*Process:* EcClWbJdt *Class:* JDTApplet	A user submits a request to the **EcCsMojoGateway** to have ASTER data taken (a data acquisition request or DAR) using the parameters entered into the Java DAR Tool. DAR parameters are required for submittal of DARs as specified in the ASTER GDS IRD/ICD.
Submit ASTER DAR Search	One per set of DAR parameters or DAR ID	*Process:* EcCsMojoGateway *Class:* EcMjDarQueryxARScenesProxy	*Process:* EcClWbJdt Class: JDTApplet	The EcClWbJdt process sends, to the **EcCsMojoGateway**, the request to search the ASTER DAR database by DAR parameters or a specific DAR ID for a scene of interest (to the user) from the ASTER instrument.
Send DAR ID	One per set of DAR parameters	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* EcCsMojoGateway Class: EcMjDarQueryxARScenesProxy	As the result of a successfully submitted DAR, the EcClWbJdt receives a DAR ID from the **EcCsMojoGateway**. This is a string of characters used to track a DAR. The user receives notification every time data resulting from this DAR is received by the ECS.
Return Session Id	One per user request	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* EcCsMojoGateway Class: EcMjDCELoginProxy	The **EcCsMojoGateway** returns a session id for the user to communicate between the EcClWbJdt (DAR Tool) and the ASTER GDS via the EcGwDARServer.
Send Subscription Notification	One per subscription submitted	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* EcCsMojoGateway *Class:* EcMjECSSbsrvProxy	The **EcCsMojoGateway** returns the subscription notification to the EcClWbJdt for the user.

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (4 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Subscription	One per notification request	*Process:* EcSbSubServer *Library:* EcSbCl *Class:* EcClSubscription	*Process:* EcCsMojoGateway *Class:* EcMjECSSbsrvProxy	The EcCsMojoGateway passes a subscription request to the EcSbSubServer on behalf of a user. This is a request for notification upon a specific event occurring in the system. An example would be subscribing to the insert of a particular granule type. A valid subscription request results in the return of a subscription identifier. The subscription Identifier is not returned to the user.
Notify of Subscription	One per subscription submitted	*Process:* EcCsMojoGateway *Class:* EcMjECSSbsrvProxy	*Process:* EcSbSubServer *Library:* EcSbSr *Classes:* EcSbSubscription, EcSbNotification	The EcSbSubServer sends E-mail to the ECS User, Operations Staff (via EcSbGui), or inter-process notification (via the message-passing framework) to the EcCsMojoGateway to notify the user when his/her subscription criteria have been met.
Submit/Modify DAR	One per request to DAR	*Process:* EcGwDARServer *Library:* EcGwDAR *Classes:* EcGwDARSubmitDarRequest_C EcGwDARModifyDarRequest_C	*Process:* EcCsMojoGateway *Library:* EcCsMojoGateway *Classes:* EcMjDarSubmitDarProxy, EcMjDarModifyDarProxy	The EcCsMojoGateway submits the DAR Submit request (and all other DAR related requests) after selecting or modifying DAR parameters to the **EcGwDARServer**. The EcGwDARServer interfaces directly with the ASTER GDS, and, in the event of a DAR Submit Request, returns a DAR ID to the EcCsMojoGateway. The EcCsMojoGateway, in turn, returns the DAR ID to the EcClWbJdt. The EcCsMojoGateway handles all ECS service requests from the EcClWbJdt via proxies.

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (5 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Search ASTER DAR Database	One per set of DAR parameters or DAR ID	*Process:* EcGwDARServer *Library:* EcGwDAR *Class:* EcGwDARQueryxARScenesRequest_C	*Process:* EcCsMojoGateway *Library:* EcCsMojoGateway *Class:* EcMjDarQueryxARScenesProxy	The EcCsMojoGateway submits a request to the **EcGwDARServer** to search the ASTER GDS DAR database for DARs and their respective status (i.e., acquired scenes). Search qualifications may be in the form of DAR parameters or DAR IDs. To get a status of the search, users may view the Search Status via the Java DAR Tool.
Return DAR ID	One per DAR submitted	*Process:* EcCsMojoGateway *Library:* EcCsMojoGateway *Class:* EcMjDarQueryxARScenesProxy	*Process:* EcGwDARServer *Library:* EcGwDAR *Class:* EcGwDARQueryxARScenesRequest_C	As the result of a successfully submitted DAR, the EcCsMojoGateway receives a DAR ID from the EcGwDARServer. The DAR ID is a string of characters used to track a DAR.
Request Management Services (RMS)	One per service request	N/A	N/A	The **EcMsAcRegUserSrvr** provides a basic management library of services to the processes, implemented as client or server applications, using the DCCI CSCI Process Framework. The basic management library of services includes the items listed below.
RMS (cont.)	One per notice received	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Classes:* MsAcUsrProfile, RWPortal	*Process:* EcCsMojoGateway *Class*: EcCsRetrieveProfileProxy	**User Profile Request** – The **EcMsAcRegUserSrvr** provides requesting processes with User Profile parameters such as e-mail and shipping addresses to support their processing activities.

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (6 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Return User Profile	One per request	*Process:* EcCsMojoGateway *Class*: EcCsRetrieveProfileProxy	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Classes:* MsAcUsrProfile, RWPortal	The **EcMsAcRegUserSrvr** returns the user profile to the EcCsMojoGateway.
Export Location Information	Once per session	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	*Process:* EcCsMojoGateway *Library:* EcCsMojoGateway *Class:* EcMjManagedSrv	The EcCsMojoGateway sends physical and logical server location information to the **EcCsIdNameServer** to locate ECS data.
Import Location Information	One per user request	*Process:* EcCsMojoGateway *Library:* EcCsMojoGateway *Class:* EcMjManagedSrv	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	The **EcCsIdNameServer** returns physical and logical server information to the EcCsMojoGateway.

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (7 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Login	Once per session	*Process:* Web Browser (COTS)	User	The EcClWbJdt receives the user name and password from the **user** via a web browser and requests a user profile from the EcCsMojoGateway to authenticate the user. The CCS NameServer sends back a session id.
Enter DAR Parameters	One set per acquisition	*Process:* Web Browser (COTS)	User	A **user** enters parameters (as specified in the Interface Control Document (ICD)) required for submittal or modification of Data Acquisition Requests (DARs) in accordance with the ASTER GDS Interface Requirements Document (IRD). Upon completion of the selection or modification of DAR parameters, the user may submit a DAR. In addition, the user may specify DAR parameters for a search of the DAR database.
Enter Service Requests	Per user request	*Process:* Web Browser (COTS)	User	A **user** enters service requests via the web browser.
Request On-Demand Products	Per user request	*Process:* Web Browser (COTS)	User	**A user** selects the On-demand Product (ASTER L1B, ASTER DEM, and ASTER higher level) and a processing parameter(s).

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (8 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Request ASTER Data Acquisition	Per user request	*Process:* Web Browser (COTS)	User	A **user** submits a request via a Web Browser to have ASTER data taken (a data acquisition request or DAR) using the parameters entered into the Java DAR Tool. DAR parameters are required for submittal of DARs as specified in the ASTER GDS IRD/ICD.
Request ASTER DAR Search	Per user request	*Process:* Web Browser (COTS)	User	A **user** submits a request via a Web Browser to search the ASTER DAR database by DAR parameters or a specific DAR ID to determine if the ASTER instrument has acquired a scene of interest (to the user).
Send Login Request	One per User	*Process:* EcClWbJdt *Class:* ClOdUserLogin	*Process:* Web Browser (COTS)	The **Web Browser** sends the user id information (name and password) to the EcClWbJdt process to obtain a session id to submit data and service requests.
Send DAR Parameters	One set per data acquisition	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* Web Browser (COTS)	The **Web Browser** sends the ASTER DAR parameters to the EcClWbJdt to get the request processed.
Send Service Requests	Per user request	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* Web Browser (COTS)	The **Web Browser** sends the user service requests to the EcClWbJdt.
Create DAR	One acquisition at a time	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* Web Browser (COTS)	The **Web Browser** sends the data acquisition request to the EcClWbJdt.
Send On-Demand Request	One request at a time	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* Web Browser (COTS)	The **Web Browser** sends the ASTER On-Demand requests to the EcClWbJdt to get the request processed.

Table 4.3-4. WKBCH CSCI (Java DAR Tool) Process Interface Events (9 of 9)

Event	Event Frequency	Interface	Initiated By	Event Description
Send ASTER DAR Search	Per user request	*Process:* EcClWbJdt *Class:* JDTApplet	*Process:* Web Browser (COTS)	A user submits a request via a **Web Browser** to search the ASTER DAR database by DAR parameters or a specific DAR ID to determine if the ASTER instrument has acquired a scene of interest (to the user).
Receive DAR ID	One per DAR submitted	***Process:*** Web Browser (COTS) User	*Process:* EcClWbJdt *Class:* JDTApplet	The **Web Browser** receives the DAR ID from the EcClWbJdt.
Receive Session Id	One per user session	***Process:*** Web Browser (COTS) User	*Process:* EcClWbJdt *Class:* JDTApplet	The **Web Browser** receives the session id from the EcClWbJdt.
Receive Subscription Notification	One per subscription submitted	***Process:*** Web Browser (COTS) User	*Process:* EcClWbJdt *Class:* JDTApplet	The **Web Browser** receives the subscription notification from the EcClWbJdt. The user receives notification every time data resulting from this DAR is received by the ECS.

4.3.1.6 Workbench Data Stores

Table 4.3-5 provides descriptions of the WKBCH CSCI data storage areas shown on the EOSView and Java DAR Tool Architecture Diagrams. To simplify the table, the list of data stores is limited to the areas shown.

Table 4.3-5. WKBCH CSCI Data Stores

Data Store	Type	Functionality
HDF File	File	A listing of the HDF files accessible by EOSView.
Session Data	File	The Java DAR Tool maintains user session data, which includes submitted and in process DARs, search criteria and search results.

4.3.2 Desktop Software Description

4.3.2.1 Desktop Functional Overview

The DESKT CSCI provides the User Profile Gateway server to communicate with the MSS User Registration Server for obtaining user profile information to authenticate users or update user information.

4.3.2.2 Desktop Context

Figure 4.3-5 is the DESKT CSCI context diagram. The diagram shows the events sent to the DESKT CSCI and events the DESKT CSCI sends to other CSCIs. Table 4.3-6 provides descriptions of the interface events shown in the DESKT CSCI context diagram.

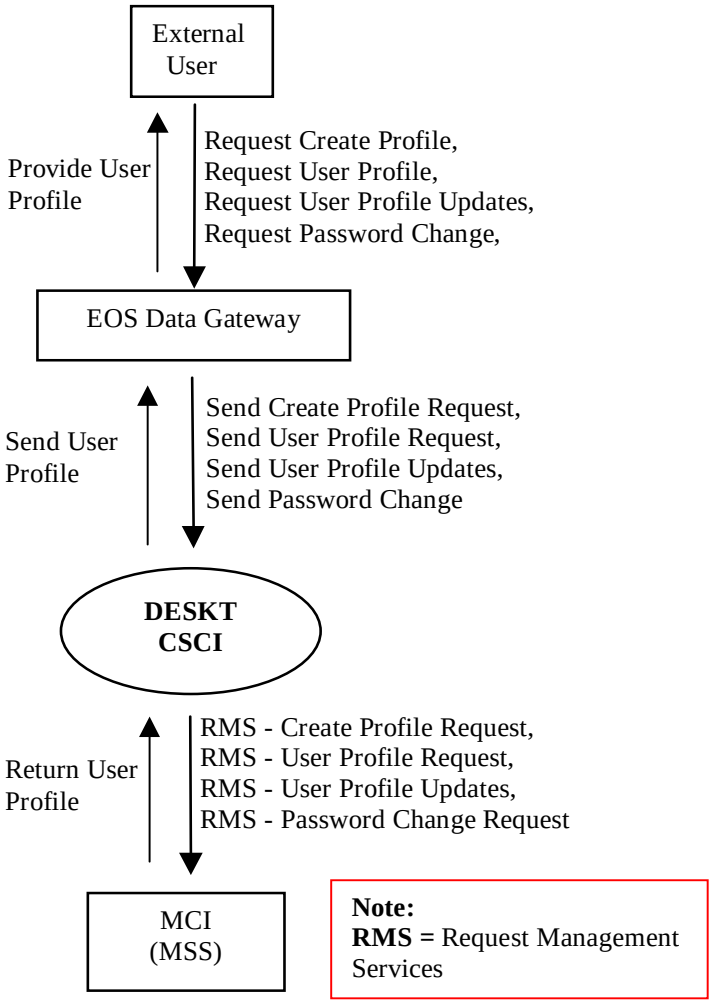


Figure 4.3-5. DESKT CSCI Context Diagram

Table 4.3-6. DESKT CSCI Interface Events

Event	Interface Event Description
Request Create Profile	A **user** submits a request to be a registered user of the ECS to the EOS Data Gateway (to forward to the MCI) to create a user profile. The user profile includes, but is not limited to, mailing, billing and shipping contact information.
Request User Profile	The **user** sends a request to the EOS Data Gateway (to the MCI) for retrieval of a User Profile. The user profile is the important information about a user that must be maintained. This includes, but is not limited to, mailing, billing, and shipping addresses, phone number, and electronic mail address.
Request User Profile Updates	**Users** can update their User Profile information through the EOS Data Gateway (EDG). This includes their addresses (user, shipping, billing, and e-mail) and other information. This file gets forwarded to the DESKT CSCI and the DESKT CSCI forwards the file to the MCI.
Request Password Change	The **user** can request a change or reset of his/her EDG/ECS User Profile passwords through the EOS DATA Gateway. The user enters his userid, current and new passwords. The userid and passwords are converted to authenticators. This information is used to change the EDG password and the authenticators (old and new) are forwarded through the CLS to the MSS. The MSS changes the ECS authenticator. The authenticator is an encrypted version of the userid and password. It is encrypted so it can be passed over a socket without threat of being stolen.
Send Create Profile Request	The **EOS Data Gateway** sends a request to the DESKT CSCI to register the user in the ECS.
Send User Profile Request	The **EOS Data Gateway** sends the user profile request to the DESKT CSCI for processing.
Send User Profile Updates	The **EOS Data Gateway** sends the user profile updates to the DESKT CSCI for processing.
Send Password Change	The **EOS Data Gateway** sends a request to the DESKT CSCI to change or reset the user's password in the ECS for user access.
Request Management Services	The **MCI** provides a basic management library of services to the CSCIs, implemented as client or server applications, using the DCCI CSCI Process Framework. The basic management library of services includes: • **Create Profile Request** – The MCI receives user information for becoming a registered user of the ECS from the DESKT CSCI. The MCI sends a response to the user when the request is received. • **User Profile Request** – The MCI provides requesting CSCIs with User Profile parameters such as e-mail address and shipping address upon request by authorized users to support their processing activities. • **User Profile Updates** – The MCI receives user profile parameter updates from a user and makes the updates in the user profile database. • **Password Change Request** – The DESKT CSCI sends requests on behalf of ECS users to the MCI to change or reset users' authenticators in the MSS database.
Return User Profile	The MSS Sybase ASE returns the user profile to the DESKT CSCI, via the **MCI**, to forward back to the user via the EOS Data Gateway.
Send User Profile	The DESKT CSCI returns the user profile to the **EOS Data Gateway**.
Provide User Profile	The EOS Data Gateway provides the user profile to the external user.

4.3.2.3 Desktop Architecture

Figure 4.3-6 is the DESKT CSCI architecture diagram. The diagram shows the events sent to the DESKT CSCI processes and the events the DESKT CSCI processes send to other processes.

The DESKT CSCI consists of one process. This process is the User Profile Gateway, a server that listens for calls on a socket. This process resides inside the DAAC on the INTHW-1 server.

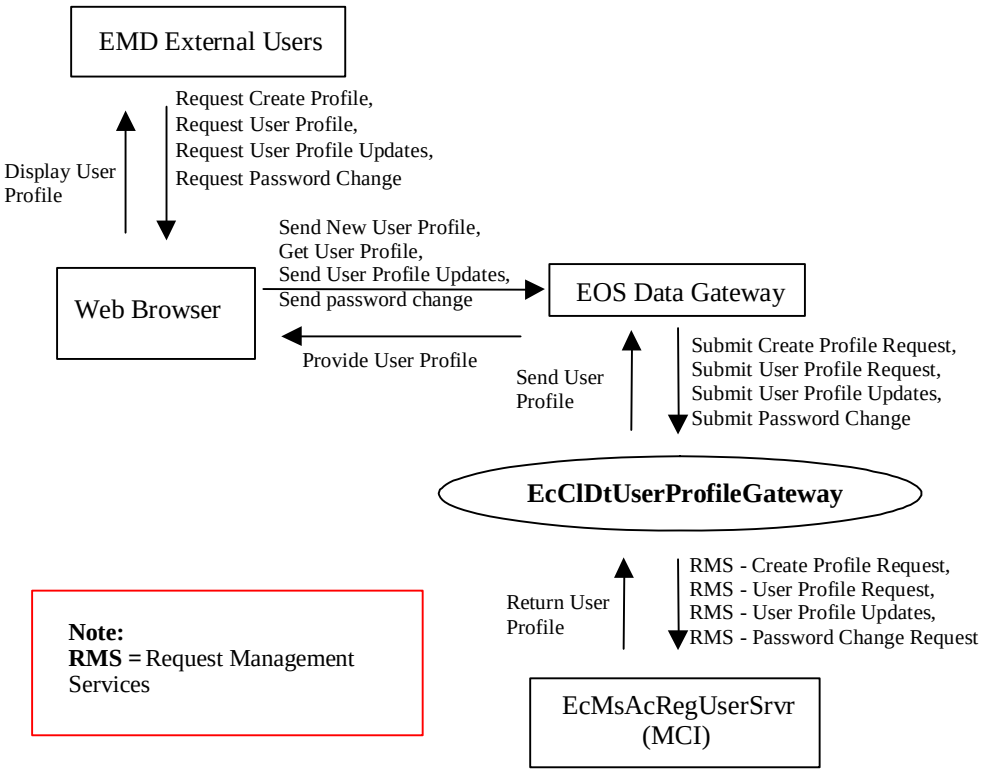


Figure 4.3-6. DESKT CSCI Architecture Diagram

4.3.2.4 Desktop Process Description

Table 4.3-7 provides descriptions of the processes shown in the DESKT CSCI architecture diagram.

Table 4.3-7. DESKT CSCI Processes

Process	Type	COTS/ Developed	Functionality
EcClDtUserProfileGateway	Server	Developed	The User Profile Gateway is a retrieval and update engine for the user profile information. The server listens for calls on a socket. Interfaces: • User Profile Request: Request for a user profile given the user's authenticator. • Profile Update Request: Request to update the user's profile with new information. Server Supports: • Multiple concurrent requests.

4.3.2.5 Desktop Process Interface Descriptions

Table 4.3-8 provides descriptions of the interface events shown in the DESKT CSCI architecture diagram.

Table 4.3-8. DESKT CSCI Process Interface Events (1 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Create Profile	Once per User	*Process:* Web Browser (COTS)	User	This is a request by a user to register (establish an account) in the ECS.
Request User Profile	One per User Login	*Process:* Web Browser (COTS)	User	The **user** sends a request to the EcMsAcRegUserSrvr using the ECS Authenticator from the EcClDtUserProfileGateway or the user's ID.
Request User Profile Updates	One per User Profile update	*Process:* Web Browser (COTS)	User	**Users** can update their profiles via the Web Browser through the EOS Data Gateway.
Request Password Change	One per password change	*Process:* Web Browser (COTS)	User	**Users** can update or reset their passwords via the Web Browser through the EOS Data Gateway.
Send New User Profile	Once per User	*Process:* EOS Data Gateway	*Process:* Web Browser (COTS)	The Web Browser sends the User Registration request to the EOS Data Gateway for submittal to the ECS.
Get User Profile	One per User Profile	*Process:* EOS Data Gateway	*Process:* Web Browser (COTS)	The Web Browser sends the request to obtain a profile to the **EOS Data Gateway**.

Table 4.3-8. DESKT CSCI Process Interface Events (2 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Send User Profile Updates	One set of parameters per request	*Process:* EOS Data Gateway	*Process:* Web Browser (COTS)	The Web Browser sends a request to update profile information to the **EOS Data Gateway**. This interface to the EcClDtUserProfileGateway requests an update to the user's profile.
Send Password Change	Once per password change	*Process:* EOS Data Gateway	*Process:* Web Browser (COTS)	The Web Browser sends a password change or reset request to the **EOS Data Gateway** to be processed within the ECS.
Submit Create Profile Request	Once per User	*Process:* EcClDtUserProfile Gateway *Class:* ClDtProfileServer	*Process:* EOS Data Gateway	The EOS Data Gateway submits the request to register a user to the EcClDtUserProfileGateway and ultimately processed by the EcMsAcRegUserSrvr within the MSS.
Submit User Profile Request	One per user request	*Process:* EcClDtUserProfile Gateway *Class:* ClDtProfileServer	*Process:* EOS Data Gateway	The **EOS Data Gateway** submits the request to obtain a profile to the EcClDtUserProfileGateway and ultimately processed by the EcMsAcRegUserSrvr.
Submit User Profile Updates	One set of parameters per request	*Process:* EcClDtUserProfile Gateway *Class:* ClDtProfileServer	*Process:* EOS Data Gateway	The **EOS Data Gateway** submits the request to update a profile to the EcClDtUserProfileGateway and ultimately processed by the EcMsAcRegUserSrvr.
Submit Password Change	Once per password change	*Process:* EcClDtUserProfile Gateway *Class:* ClDtProfileServer	*Process:* EOS Data Gateway	The **EOS Data Gateway** submits the request to change or reset a password to the EcClDtUserProfileGateway, which is ultimately processed by the EcMsAcRegUserSrvr.

Table 4.3-8. DESKT CSCI Process Interface Events (3 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Management Services (RMS)	One per service request	N/A	N/A	The **EcMsAcRegUserSrvr** provides a basic management library of services to the processes, implemented as client or server applications, using the DCCI CSCI Process Framework. The basic management library of services includes the items listed below.
RMS (cont.)	One per user request	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Classes:* MsAcUsrProfile, RWPortal	*Process:* EcClDtUserProfileGateway *Class:* ClDtProfileServer	**User Profile Request** – The **EcMsAcRegUserSrvr** provides requesting processes with User Profile parameters such as e-mail and shipping addresses to support their processing activities.
RMS (cont.)	One set of parameters per user request	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Classes:* MsAcUsrProfile, RWPortal	*Process:* EcClDtUserProfileGateway *Class:* ClDtProfileServer	**User Profile Updates** – The **EcMsAcRegUserSrvr** provides requesting processes with access to User Profile parameters such as e-mail and shipping addresses to support the update of the parameters.
RMS (cont.)	One set of parameters per user request	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Classes:* MsAcUsrProfile, RWPortal	*Process:* EcClDtUserProfileGateway *Class:* ClDtProfileServer	**Password Change Request** – The EcClDtUserProfileGateway sends requests on behalf of ECS users to the **EcMsAcRegUserSrvr** to change or reset a user's password in the MSS database. The password is represented as an authenticator in the MSS database.

Table 4.3-8. DESKT CSCI Process Interface Events (4 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Return User Profile	One per user request	*Process:* EcClDtUserProfile Gateway *Class:* ClDtProfileServer	*Process:* EcMsAcRegUserSrvr *Library:* MsAcClnt *Classes:* MsAcUsrProfile, RWPortal	The **EcMsAcRegUserSrvr** returns the user profile to the EcClDtUserProfileGateway.
Send User Profile	One per user request	*Process:* EOS Data Gateway	The EcClDtUserProfile Gateway forwards the user profile to the EOS Data Gateway	The EcClDtUserProfile Gateway forwards the user profile to the **EOS Data Gateway.**
Provide User Profile	One per user request	*Process:* Web Browser (COTS)	*Process:* EOS Data Gateway	The EOS Data Gateway returns the profile information to the Web Browser.
Display User Profile	One per user request	External Users	*Process:* Web Browser (COTS)	The Web Browser displays the user profile information to the external users.

4.3.2.6 Desktop Data Stores

There are no data stores used by the DESKT CSCI.